ON THE APPLICATION OF SUDOKU TO ERROR CORRECTION IN COMMUNICATION

An Honors Fellow Thesis

by

ANDREW JOHN YOUNG

Submitted to Honors and Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as

HONORS UNDERGRADUATE RESEARCH FELLOW

May 2012

Major: Electrical Engineering

Mathematics

ON THE APPLICATION OF SUDOKU TO ERROR CORRECTION IN COMMUNICATION

An Honors Fellow Thesis

by

ANDREW JOHN YOUNG

Submitted to Honors and Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as

HONORS UNDERGRADUATE RESEARCH FELLOW

Approved by:

Research Advisor,	Costas N. Georghiades
Director, Honors and Undergraduate Research,	Duncan MacKenzie

May 2012

Major: Electrical Engineering

Mathematics

ABSTRACT

On the Application of Sudoku to Error Correction in Communications. (May 2012)

Andrew John Young
Department of Electrical and Computer Engineering
Department of Mathematics
Texas A&M University

Research Advisor: Dr. Costas N. Georghiades
Department of Electrical and Computer Engineering

This work provides a detailed analysis of the general Sudoku problem with particular attention given to the first nontrivial case. A formal mathematical model is developed and used to determine several important parameters; including error upper bound, equivalence class decomposition, and entropy optimal path. The implicit communication problem is established and exhaustive simulation provides an explicit expression for the probability of error. Using an ordered transmission sequence or path the model is extended to rateless communication. These analyses show that a channel code using the Sudoku constraint is strictly suboptimal, but concatenation with a rateless framework shows promise.

ACKNOWLEDGMENTS

I would like to thank Dr. Georghiades for his continued support throughout the research process.

TABLE OF CONTENTS

CHAPTER Page

ABSTRACT iii

ACKNOWLEDGMENTS iv

TABLE OF CONTENTS v

LIST OF FIGURES vii

LIST OF TABLES xi

I INTRODUCTION 1

II METHODS 6

III THE SUDOKU PROBLEM 7

A. General n Sudoku constraint 7
B. The n = 2 Sudoku constraint 10
C. Equivalence classes 13

IV THE COMMUNICATION PROBLEM 17

A. Encoding 17
B. Channel model 18
C. Decoding 19
D. Error patterns and simulations 21

V PATHS 26

VI CONCLUSIONS 32

REFERENCES 33

APPENDIX A 35

APPENDIX B 72

CHAPTER Page

APPENDIX C 74

CONTACT INFORMATION 77

LIST OF FIGURES

FIGURE		Page
1	$n = 2$ Sudoku Tanner Graph	9
2	Enumerated Cases	11
3	Fully Enumerated Cases	12
4	Equivalence Class Matrices	16
5	Explicit Error Probability	25
6	Path Length Empirical PMF	29
7	An Optimal Path	30
8	1-1 1-22 1-24 1-3	35
9	1-2 1-16 1-18 1-5	35
10	1-4 1-10 1-12 1-6	36
11	1-7 1-20 1-23 1-9	36
12	1-8 1-14 1-17 1-11	36
13	1-13 1-19 1-20 1-15	36
14	2-1 3-11 2-17 3-22 2-24 2-8 3-3 3-14	37
15	2-2 3-9 2-23 3-16 2-18 2-7 3-5 3-20	37
16	2-4 3-15 2-21 3-10 2-12 2-13 3-6 3-19	38
17	3-1 2-22 3-8 2-14 3-17 3-24 2-3 2-11	38
18	3-2 2-16 3-7 2-20 3-23 3-18 2-5 2-9	39
19	3-4 2-10 3-13 2-19 3-21 3-12 2-6 2-15	39

FIGURE Page

20 4-1 4-11 4-24 4-14 4-17 4-8 4-3 4-22 40

21 4-2 4-9 4-18 4-20 4-23 4-7 4-5 4-16 40

22 4-4 4-15 4-12 4-19 4-21 4-13 4-6 4-10 41

23 1-1 2-17 3-1 2-1 3-8 4-17 1-17 4-24 4-1 1-8 4-8 3-17 2-24 3-24 2-8 1-24 42

24 1-2 2-23 3-2 2-2 3-7 4-23 1-23 4-18 4-1 1-7 4-7 3-23 2-18 3-18 2-7 1-18 43

25 1-3 2-11 3-3 2-3 3-14 4-11 1-11 4-22 4-3 1-14 4-14 3-11 2-22 3-22 2-14 1-22 44

26 1-4 2-21 3-4 2-4 3-13 4-21 1-21 4-12 4-4 1-13 4-13 3-21 2-12 3-12 2-13 1-12 45

27 1-5 2-9 3-5 2-5 3-20 4-9 1-9 4-16 4-5 1-20 4-20 3-9 2-16 3-16 2-20 1-16 46

28 1-6 2-15 3-6 2-6 3-19 4-15 1-15 4-10 4-6 1-19 4-19 3-15 2-10 3-10 2-19 1-10 47

29 5-1 11-21 5-15 11-22 6-24 6-20 9-3 9-13 48

30 5-2 11-15 5-21 11-16 6-18 6-13 9-5 9-19 48

31 5-4 11-9 5-23 11-10 6-12 6-7 9-6 9-20 49

32 6-1 9-12 6-6 9-22 5-24 5-10 11-3 11-4 49

33 6-2 9-10 6-4 9-16 5-18 5-12 11-5 11-6 50

34 6-8 9-4 6-10 9-14 5-17 5-6 11-11 11-12 50

35 9-1 6-22 9-7 6-20 11-23 11-24 5-3 5-9 51

36 9-2 6-16 9-8 6-14 11-17 11-18 5-5 5-11 51

37 9-9 6-23 9-15 6-21 11-19 11-20 5-7 5-13 52

38 11-1 5-22 11-2 5-16 9-18 9-24 6-3 6-5 52

FIGURE Page

39 11-7 5-20 11-8 5-14 9-17 9-23 6-9 6-11 53

40 11-13 5-19 11-14 5-8 9-11 9-21 6-15 6-17 53

41 7-1 12-21 8-10 10-13 7-17 8-19 10-3 12-12 54

42 7-2 12-15 8-12 10-20 7-23 8-13 10-5 12-10 54

43 7-4 12-9 8-18 10-19 7-21 8-7 10-6 12-16 55

44 8-1 10-12 7-19 12-14 8-17 7-10 12-4 10-21 55

45 8-2 10-10 7-13 12-20 8-23 7-12 12-5 10-15 56

46 8-4 10-16 7-7 12-19 8-21 7-18 12-6 10-9 56

47 10-1 7-11 12-18 8-20 12-23 10-8 7-3 8-16 57

48 10-2 7-9 12-24 8-14 12-17 10-7 7-5 8-22 57

49 10-4 7-15 12-22 8-8 12-11 10-13 7-6 8-24 58

50 12-1 8-11 10-23 7-16 10-18 12-8 8-3 7-20 58

51 12-2 8-9 10-17 7-22 10-24 12-7 8-5 7-14 59

52 12-4 8-15 10-11 7-24 10-22 12-13 8-6 7-8 59

53 5-1 6-17 7-1 6-1 8-8 8-17 5-17 7-24 8-1 6-8 7-8 7-17 5-24 8-24 5-8 6-24 60

54 5-2 6-23 7-2 6-2 8-7 8-23 5-23 7-18 8-2 6-7 7-7 7-23 5-18 8-18 5-7 6-18 61

55 5-3 6-11 7-3 6-3 8-14 8-11 5-11 7-22 8-3 6-14 7-14 7-11 5-22 8-22 5-14 6-22 62

56 5-4 6-21 7-4 6-4 8-13 8-21 5-21 7-12 8-4 6-13 7-13 7-21 5-12 8-12 5-13 6-12 63

57 5-5 6-9 7-5 6-5 8-20 8-9 5-9 7-16 8-5 6-20 7-20 7-9 5-16 8-16 5-20 6-16 64

58 5-6 6-15 7-6 6-6 8-19 8-15 5-15 7-10 8-6 6-19 7-19 7-15 5-10 8-10 5-19 6-10 65

FIGURE Page

59 9-1 12-17 11-1 10-1 9-8 10-17 11-17 10-24 12-1 10-8 12-8 9-17 12-24 9-24 11-8 11-24 66

60 9-2 12-23 11-2 10-2 9-7 10-23 11-23 10-18 12-2 10-7 12-7 9-23 12-18 9-18 11-7 11-18 67

61 9-3 12-11 11-3 10-3 9-14 10-11 11-11 10-22 12-3 10-14 12-14 9-11 12-22 9-22 11-14 11-22 68

62 9-4 12-21 11-4 10-4 9-13 10-21 11-21 10-12 12-4 10-13 12-13 9-21 12-12 9-12 11-13 11-12 69

63 9-5 12-9 11-5 10-5 9-20 10-9 11-9 10-16 12-5 10-20 12-20 9-9 12-16 9-16 11-20 11-16 70

64 9-6 12-15 11-6 10-6 9-19 10-15 11-15 10-10 12-6 10-19 12-19 9-15 12-10 9-10 11-19 11-10 71

LIST OF TABLES

TABLE Page

I Permutation Ordering 13

II Subgroup Decomposition 16

III Symmetric Error Matrix 22

IV Asymmetric Error Matrix 23

CHAPTER I

INTRODUCTION

The primary medium for storage, transfer, and conveyance of information has transitioned from physical to digital. New techniques have been developed to ensure reliability and security of this information. Coding is one such safeguard that has been widely adopted for a variety of important applications. Some of the more common uses of coding include error correction, compression, and encryption.

Error-correcting codes use redundant information to alleviate the affects of noise and interference. An assortment of coding schemes exists and their respective performance is highly dependent on the desired application. This performance dependence is so profound that some codes were developed for use with only one particular application.

A more recent example of this is product codes. Recent advances in optical communication, particularly in the form of fiberoptic transmission cables, provide communication systems with very high data rates, on the order of 10 Gbps. To code at these extremely high rates requires very low decoding complexity, and product codes provide this. A product code imposes two-dimensional code constraints on an array of information bits. The simplest case is where every row and column must sum to zero modulo two. While there are other codes that provide superior error correction with higher decoding complexity, product codes have found their niche and are a common tool in optical communication.

This thesis follows the style of *IEEE Transactions on Communications*.

At the most basic level digital information is a sequence of 1's and 0's. For example, each of the letters in this text is represented by an 8 bit binary ASCII code, i.e. $A = 01000001$. While this representation may seem convoluted, its overall function is rather simple, to convey information, and binary encoding provides a practically implementable map. Storage of information is important, but communication of that information is of equal if not greater value.

Consider the simple task of transmitting one bit of information, either yes represented by 1, or no represented by 0. In a perfect digital world either a 1 or a 0 would be transmitted from an antenna and then received. However, the world is not digital it is analog thus 1's and 0's must be converted into waveforms. During their journey these waveforms will experience distortion from noise, attenuation, etc, and who is to say whether a no is received as a yes or vice versa. In most cases inaccurate information is useless information, thus steps must be taken to insure reliability.

One popular method to increase the accuracy of information transfer is the addition of redundant information. The simplest coding scheme involves sending the same message multiple times. For example 111 will denote yes and 000 will denote no, and the "correct" message will be determined by whether more 0's or 1's are received, i.e. 101 is a yes and 001 is a no. Using this scheme all single bit errors can be corrected. However, the trade of comes in the fact that a three bit message is used to transmit a single bit of information. Herein lies the fundamental trade off in coding between accuracy and rate. The content of the information being sent will determine whether one is to be favored over the other, i.e. financial data will probably favor accuracy whereas streaming music would favor rate.

This work concentrates on an unconventional channel coding technique utilizing the underlying constraint equations of the sudoku puzzle. Sudoku is a popular numerically based puzzle found in newspapers and magazines. The typical game involves placement of the numbers $\{1, .., 9\}$ in 9 different 3×3 sub blocks forming a 9×9 block under the constraint that each row, column, and 3×3 sub block may contain only one instance of each number. It has been shown through exhaustive simulation that there are $6,670,903,752,021,072,936,960$ possible valid puzzles, but only $5,472,730,538$ of these are unique up to isomorphism [1].

The common everyday Sudoku puzzle is a specific example in a more general group of constraint problems. The general model involves a collection of n^2 unique elements; a common construct is the first n^2 natural numers, $1, \ldots, n^2$. A selection of n^4 of these elements are chosen to comprise the entries of an $n^2 \times n^2$ block array. This block is partitioned into n^2 subblocks, rows, and columns; with the added constraint that each element is uniquely present within one of these subpartitions.

Sudoku's popularity in Western Europe and the Americas was minimal until the mid 2000's, but its origins date back much further. Leonard Euler introduced a similar notion, the latin square, in the 1740's. The latin square is an $m \times m$ array filled with m different latin letters under the constraint that each row and column may contain only one instance of each of the m letters, sound familiar. Creation of the modern day sudoku is credited to Howard Garns, a retired architect and avid puzzle designer. His puzzle model was published in a 1979 issue of Dell Magazine under the name Number Place but was given little interest. It wasn't until 1984, when the puzzle emerged in Japan, that its popularity began to grow. The puzzle was introduced by the Japanese puzzle company Nikoli in the April issue of Monthly Nikolist under the name Sũji wa

dokushin ni kagiru, "the digits must be single", later abreviated to sudoku. In fall 2004 The Times, a London Newspaper, began publishing sudoku puzzles daily, and soon after the puzzles spread throughout Western Europe and across the Atlantic [2].

Sudoku puzzles come in varying difficulties, ranging from very easy to very hard. The metric for determining the difficulty of a particular puzzle is highly correlated with the percentage of squares initially filled in. Given a 9×9 sudoku puzzle there are 81 possible entries, and a particular puzzle will be labeled according to the number of entry values given and denoted a k-clue puzzle. In general for $k < j$ a k-clue puzzle is more difficult to solve then a j-clue puzzle, i.e. a 20-clue puzzle is much more difficult then a 70-clue. Much work has gone into determining the minimum solvable, to a unique final set, puzzle. So far several 17-clue puzzles have been found, but it is unsure as to whether or not a 16-clue exists.

The sudoku puzzle belongs to the broad class of constraint satisfaction problems, and furthermore finding a solution is an NP-complete problem. This class of problems has extensive applications in the field of computational computer science, and is a millennium prize problem, sometimes abbreviated $P = NP$. The question of whether $P = NP$ involves the computational time required to solve a problem. NP denotes nondeterministic polynomial time, meaning the computational time for the algorithm cannot be upper-bounded by a polynomial equation. The $P = NP$ question seeks to determine whether a problem, like the sudoku puzzle, for which a valid solution can be checked in polynomial time can be solved in polynomial time.

From a coding perspective, the sudoku puzzle provides error robustness in its ability to reconstruct the entirety of information from a limited data set. The error suscepti-

bility of a sequence of information bits can be alleviated by mapping data to a sudoku puzzle prior to transmission. The unique structure of the sudoku puzzles should provide increased reliability in difficult coding environments; such as deep fading where transmissions are prone to large blocks of erasures and errors.

Communication of information is a fundamental need that crosses all boundaries of time and civilization. While the medium and content may change, the basic principle and underlying objectives are the same. Today the primary medium is digital information, and much work has gone into developing efficient methods of communication. This work seeks to develop such a communication scheme using the constraining equations of the sudoku puzzle.

CHAPTER II

METHODS

The proceeding analysis requires a sufficient background in mathematics and communication theory. Using these techniques the Sudoku constraint is formalized and a variety of important parameters are developed. For brevity, a list of important topics is provided

- Set Theory - Sets, subsets, and equivalence classes [3]
- Abstract Algebra - Groups and subgroups [3]
- Probability Theory - Discrete random variables [4]
- Digital Communication - Channel coding and communication channels [5, 6]
- Information Theory - Entropy [7]

Set theoretic and algebraic ideas are used to classify the set of Sudoku constrained arrays. Ideas from probability and coding theory are used to analyze the viability of an error correcting code using Sudoku. A variety of computer simulations are performed, and thusly, appropriate computational software is required. Matlab is used but any equivalent computational software will suffice.

CHAPTER III

THE SUDOKU PROBLEM

A. General n Sudoku constraint

Consider an n^2 dimensional discrete alphabet $\mathcal{X}(n)$. Without loss of generality assume $\mathcal{X}(n)$ is the first n^2 integers, $\mathcal{X}(n) = N_{n^2} = \{1, 2, \ldots, n^2\}$. Let $X \in N_{n^2}^{n^2 \times n^2}$, an n^2 by n^2 array drawn entry wise from the collection N_{n^2}. The elements of X are given the canonical enumeration

$$X = \begin{bmatrix} x_1 & x_2 & \ldots & x_{n^2} \\ x_{n^2+1} & x_{n^2+2} & \ldots & x_{n^4} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(n^2-1)n^2+1} & x_{(n^2-1)n^2+2} & \ldots & x_{n^4} \end{bmatrix}. \tag{1}$$

Partition X into n^2 blocks, rows, and columns,

$$\begin{aligned} X &= \begin{bmatrix} \mathbf{b}_1(X) & \mathbf{b}_2(X) & \ldots & \mathbf{b}_n(X) \\ \mathbf{b}_{n+1}(X) & \mathbf{b}_{n+2}(X) & \ldots & \mathbf{b}_{2n}(X) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{b}_{(n-1)n+1}(X) & \mathbf{b}_{(n-1)n+2}(X) & \ldots & \mathbf{b}_{n^2}(X) \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1(X) \\ \mathbf{r}_2(X) \\ \vdots \\ \mathbf{r}_{n^2}(X) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{c}_1(X) & \mathbf{c}_2(X) & \ldots & \mathbf{c}_{n^2}(X) \end{bmatrix}. \end{aligned} \tag{2}$$

Let $B_k(X)$, $R_k(X)$, and $C_k(X)$ be the unordered sets of elements for partitions $\mathbf{b}_k(X)$, $\mathbf{r}_k(X)$, and $\mathbf{c}_k(X)$, respectively,

$$P_i(X) = \{x \in X \mid x \in \mathbf{p}_i(X)\}. \tag{3}$$

Let $f_n : N_{n^2} \to \{0, 1\}$ be the canonical set indicator function where

$$f_n(P) = \begin{cases} 1 & P = N_{n^2} \\ \\ 0 & \text{else} \end{cases}. \tag{4}$$

Let $\mathcal{P}(X) = \{B_1(X), \ldots, B_{n^2}(X), R_1(X), \ldots, R_{n^2}(X), C_1(X), \ldots, C_{n^2}(X)\}$, and $F_n : N_{n^2}^{n^2 \times n^2} \to \{0, 1\}$, where

$$F_n(X) = \prod_{P \in \mathcal{P}(X)} f_n(P). \tag{5}$$

Definition: An element $X \in N_{n^2}^{n^2 \times n^2}$ is Sudoku constrained if $F_n(X) = 1$.

Definition: The nth order Sudoku collection, S_n, is the set of Sudoku constrained elements of $N_{n^2}^{n^2 \times n^2}$,

$$S_n = F_n^{-1}(1) = \{X \in N_{n^2}^{n^2 \times n^2} \mid F_n(X) = 1\}. \tag{6}$$

These functional constraints are graphical depicted in Figure 1 as a generalized tanner graph where the squares represent a particular f_n.

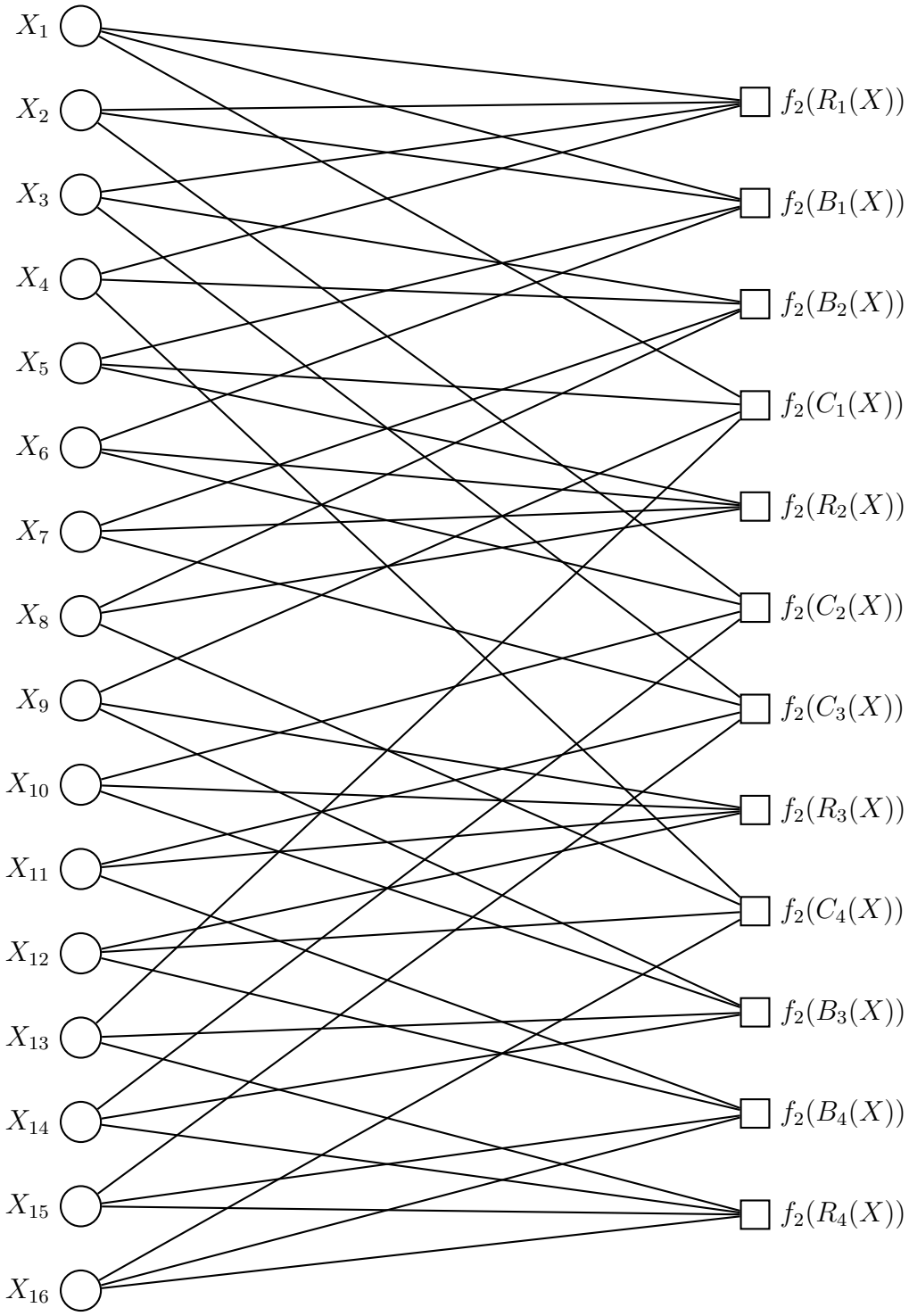


Fig. 1. $n = 2$ Sudoku Tanner Graph

B. The n = 2 Sudoku constraint

Let $n = 2$, hence $\mathcal{X}(2) = N_4 = \{1, 2, 3, 4\}$. Consider the construction of a Sudoku constrained array X. Begin with the first block, $\mathbf{b}_1(X)$. The only constraint is $B_1(X) = N_4$, where $|N_4| = 4$ and hence there are 4! possible arrangements for $\mathbf{b}_1(X)$. Without loss of generality assume

$$\mathbf{b}_1(X) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \tag{7}$$

which induces the following constraint on the entries of X

$$X = \begin{bmatrix} 1 & 2 & \{3,4\} & \{3,4\} \\ 3 & 4 & \{1,2\} & \{1,2\} \\ \{2,4\} & \{1,3\} & \{1,2,3,4\} & \{1,2,3,4\} \\ \{2,4\} & \{1,3\} & \{1,2,3,4\} & \{1,2,3,4\} \end{bmatrix}. \tag{8}$$

Next consider $\mathbf{r}_1(X)$ and $\mathbf{r}_2(X)$, there are 2 possibilities for each of the remaining entries and therefore $2! \times 2! = 4$ total combinations. Similarly for $\mathbf{c}_1(X)$ and $\mathbf{c}_2(X)$ there are 2 possibilities for each of the remaining entries and therefore $2! \times 2! = 4$ total combinations. Figure 2 enumerates these possibles. It should be noted that while all 16 of these combinations are theoretically possible they may not result in a Sudoku constrained array. In such a case an $\times$ will represent violated constraints and ?'s for the resulting undetermined entries.

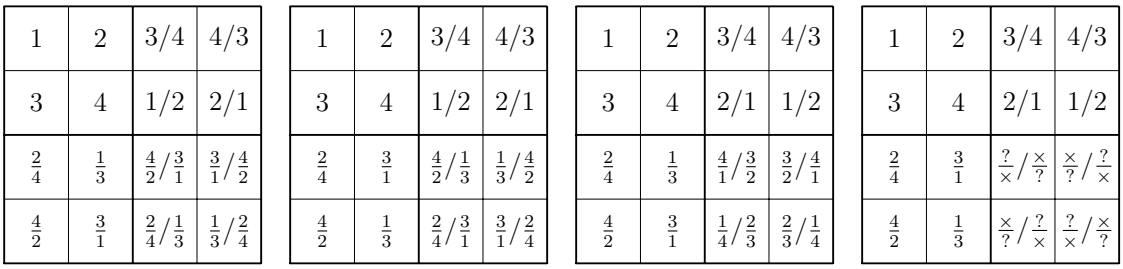


Fig. 2. Enumerated Cases

The combinations are enumerated in order of decreasing symmetry. Symmetry is defined in terms of the sub columns of $\mathbf{b}_2(X)$ with respect to $\mathbf{b}_1(X)$ and the sub rows of $\mathbf{b}_3(X)$ with respect to $\mathbf{b}_1(X)$. $\mathbf{b}_2(X)$ is said to exhibit symmetry with $\mathbf{b}_1(X)$ if

$$B_2(X) \cap C_3(X) = B_1(X) \cap C_1(X) \text{ and } B_2(X) \cap C_4(X) = B_1(X) \cap C_2(X) \quad (9)$$
$$\text{or } B_2(X) \cap C_4(X) = B_1(X) \cap C_1(X) \text{ and } B_2(X) \cap C_3(X) = B_1(X) \cap C_2(X),$$

similarly $\mathbf{b}_3(X)$ is said to exhibit symmetry with $\mathbf{b}_1(X)$ if

$$B_3(X) \cap R_3(X) = B_1(X) \cap R_1(X) \text{ and } B_3(X) \cap R_4(X) = B_1(X) \cap R_2(X) \quad (10)$$
$$\text{or } B_3(X) \cap R_4(X) = B_1(X) \cap R_1(X) \text{ and } B_3(X) \cap R_3(X) = B_1(X) \cap R_2(X).$$

Under this definition the first set is $\mathbf{b}_2(X)$ and $\mathbf{b}_3(X)$ symmetric, the second set is $\mathbf{b}_2(X)$ symmetric, the third set is $\mathbf{b}_3(X)$ symmetric, and the fourth set is neither. It should be noted that this asymmetry is translated to the fourth block through the other two.

Observing Figure 2, of the original 16 possible combinations only 12 admit valid Sudoku constrained configurations. These 4 problem configurations result from those cases with neither $\mathbf{b}_2(X)$ or $\mathbf{b}_3(X)$ symmetry. Figure 3 fully enumerates the twelve valid configurations.

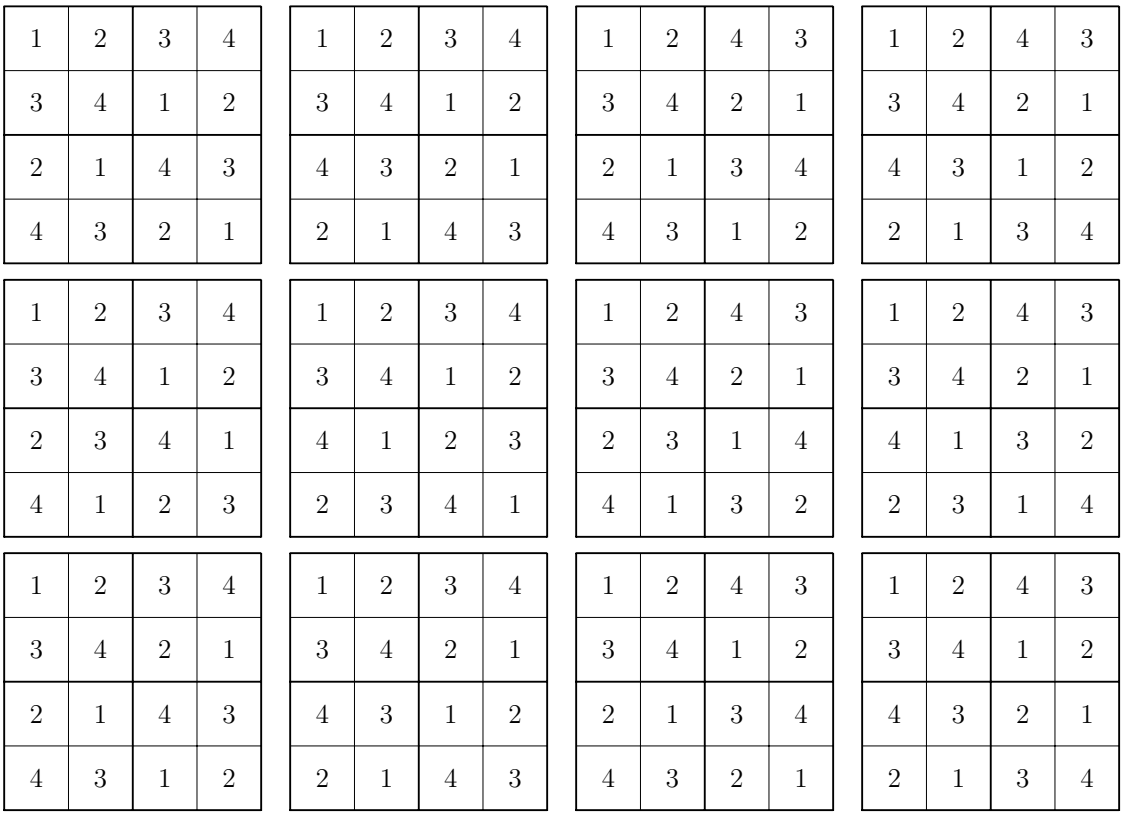


Fig. 3. Fully Enumerated Cases

This provides a total of $4! \cdot 12 = 288$ 2nd order constrained elements, $S_2 = 288$. These twelve arrays will be denoted by symmetry class 1-12, respectively, following a row column labeling convention beginning with the upper left corner. Let P_n denote the set of permutations of n, where $|P_n| = n!$. Consider the canonical ordering of these $n!$ elements. An element $p_n \in P_n$ is assigned a number between 1 and $n!$ based on its relative ordering as a number given by the order of elements in the set. The case for P_4 is depicted in Table I

Table I. Permutation Ordering

1234	1	2134	7	3124	13	4123	19
1243	2	2143	8	3142	14	4132	20
1324	3	2314	9	3214	15	4213	21
1342	4	2341	10	3241	16	4231	22
1423	5	2413	11	3412	17	4312	23
1432	6	2431	12	3421	18	4321	24

Hence a particular Sudoku constrained element will be denoted by a tuple (i, j) where $i \in \{1, 2, \ldots, 24\}$ denotes the permutation of the first block and $j \in \{1, 2, \ldots, 12\}$ denotes the corresponding symmetry class.

C. Equivalence classes

The algebraic and combinatorical nature of the Sudoku constraint arrays leads one to define a set of equivalence classes on a Sudoku collection. Two Sudoku constrained arrays $X, X' \in S_n$ are said to be equivalent, $X \sim X'$, if they are connected by a finite collection of Sudoku constrained operations, $\{\delta_k \mid k = 1, \ldots, m\}$, where

$$X' = (\delta_m \circ \ldots \circ \delta_1)(X), \tag{11}$$

and

$$(\delta_k \circ \ldots \circ \delta_1)(X) \in S_n \qquad \text{for } k = 1, \ldots, m. \tag{12}$$

In other words, two arrays are equivalent if they are connected by a set of transformations within the set of Sudoku constrained arrays.

The following operators will be used to construct these equivalence classes.

Permutation mappings, $\{\sigma_k \mid k = 1, \ldots, n^2!\}$, labeled according to the permutation ordering described above mapped with respect to the identity element $(1, 2, \ldots, n^2)$, i.e.

$$\sigma_1((1, 2, \ldots, n^2)) = (1, 2, \ldots, n^2) \tag{13}$$
$$\sigma_{n^2!}((1, 2, \ldots, n^2)) = (n^2, n^2 - 1, \ldots, 1).$$

The next two set of operators result from the conditions for a Sudoku constrained array.

Reordering rows and columns of blocks. The n^2 block equations can be further partitioned into n block row equations and n block column equations. Reordering and relabeling these block and row columns preserves a Sudoku constrained array. There are n rows and columns and thusly $n!$ possible orientations for these rows and columns. The possible transformations will be given by the following set of operators $\{\delta_i^{(k)} \mid i = 1, \ldots, n!\ k = 1, 2\}$, where k denotes row or column, 1 or 2, respectively, and i denotes the corresponding permutation mapping.

Reordering row or column equations within row or column block equations. Within a collection of block row equations or block columns equations, reordering the row and column equations, respectively, preserves Sudoku constraint. Within each row or column of blocks there are n row or block equations, providing $n!$ possible combinations. These possible transformations will be given by the following set of operator $\{\rho_{i,j}^{(k)} \mid i = 1, \ldots, n\ j = 1, \ldots, n!\ k = 1, 2\}$, where k denotes row or column, 1 or 2, respectively, i denotes the particular number of the row or column, and j denotes the

corresponding permutation mapping.

The final operator, μ, is a rotation, or rotating the array clockwise, i.e.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}. \tag{14}$$

This is the only operator that effects the blocks, rows, and columns in unison and together. This operator relabels the block equations, and interchanges sets of row and column equations in such a way that preserves Sudoku constraint.

The corresponding equivalence class decomposition for S_2 is enumerated. For $n = 2$ the following collection of operators is provided

$$\left\{\sigma_1 \quad \dots \quad \sigma_{24}\right\} \quad \left\{\delta_1^{(1)} \quad \delta_2^{(1)} \quad \delta_1^{(2)} \quad \delta_2^{(2)}\right\} \tag{15}$$

$$\left\{\rho_{1,1}^{(1)} \quad \rho_{1,2}^{(1)} \quad \rho_{2,1}^{(1)} \quad \rho_{2,2}^{(1)} \quad \rho_{1,1}^{(2)} \quad \rho_{1,2}^{(2)} \quad \rho_{2,1}^{(2)} \quad \rho_{2,2}^{(2)}\right\} \quad \left\{\mu\right\}.$$

In this case concatenation of the δ and μ operators induces transformations equivalent to the D_4 group. Combining these operations is also somewhat natural since they deal with with large scale transformations of an array as opposed to the small scale transformations of the ρ's.

Using all four of these operations partitions the total collection of Sudoku constrained arrays into two groups, the symmetric and the asymmetric group. Within these groups there are distinct subgroups based on a further restriction of the operations. In particular two sets of subgroups are considered, namely those generated by the concatenation of the δ's and μ denoted Flip Rotate, and those generated by the ρ's

denoted Row Column Interchange.

This decomposition provides some interesting results in regards to the size of these subgroups and the associated elements. Table II captures those results.

Table II. Subgroup Decomposition

	size of subgroup	number of subgroup	symmetry classes
S-FR-1	4	6	1
S-FR-2	8	6	2,3
S-FR-3	8	3	4
S-RCI	16	6	1,2,3,4
A-FR-1	8	12	5,6,9,11
A-FR-2	8	12	7,8,10,12
A-RCI-1	16	6	5,6,7,8
A-RCI-2	16	6	9,10,11,12

Appendix A enumerates these subgroup partitions in their entirety. Interestingly enough, these equivalence classes partition the space into two distinct groups, the symmetric and asymmetric. Figure 4 depicts the two arrays to be used as representatives for this decomposition.

1	2	3	4
3	4	1	2
2	1	4	3
4	3	2	1

(a) Symmetric

1	2	4	3
3	4	1	2
2	1	3	4
4	3	2	1

(b) Asymmetric

Fig. 4. Equivalence Class Matrices

CHAPTER IV

THE COMMUNICATION PROBLEM

A. Encoding

Consider the transmission of a k bit information sequence over a noisy channel. Channel coding is often used to alleviate these noise effects. A codeword alphabet $\mathcal{C}$ is mapping from the original k bit information sequence to a collection of symbols or sequences, usually providing additional redundancy. One common example are block codes which map k bits sequences to n bit sequences for $n \geq k$, providing $n-k$ bits of redundancy. These codes are often described by a $n-k \times n$ parity check matrix H, where the set of codewords corresponds to the nullspace of H^T.

We consider a codeword alphabet $\mathcal{C}$ equal to the set of Sudoku constrained arrays, S_n. That is, our information sequence will be mapped to a particular Sudoku constrained array. The benefit of this arises from the conditions for an array to be Sudoku constrained. In particular, knowledge of a subset of the array provides a great deal of information about the remaining unknown entries. Every entry has three constraint equations or f_n's, and thus knowledge of a single entry has a first order effect on $3n^2-2n$ other entries, and an additional second order effect on another $2n(n-1)^2$ entries.

Any mapping from information sequences to codewords must be injective and therefore an upper bound on the number of possible information sequences is the cardinality of the Sudoku constrained arrays or $|S_n|$. To send a particular array over a channel, each symbol must be mapped into a corresponding bit sequences. This provides an

information rate of

$$r = \frac{\log|S_n|}{n^4 \log n^2}. \tag{16}$$

However, in practice there is an intrinsic integer constraint on these information sequences providing an actual rate of

$$R = \frac{\lfloor \log |S_n| \rfloor}{\lceil n^4 \log n^2 \rceil}. \tag{17}$$

B. Channel model

Each entry in an array is assumed to be erased i.i.d. with probability ε. This memoryless symbol erasure channel is described by the collection $(\mathcal{X}(n), p(y|x), \mathcal{Y}(n))$, where

$$p(y|x) = \begin{cases} 1-\varepsilon & y = x \\ \\ \varepsilon & y = ? \end{cases}, \tag{18}$$

$\mathcal{X}(n) = \{1, \ldots, n^2\}$, and $\mathcal{Y}(n) = \{\mathcal{X}(n), ?\}$. Hence, there is no ambiguity and each entry is either known completely or unknown. This model provides a binomial random variable for the total number of erasures. Let $\mathcal{E}$ be a random variable representing the number of erased entries, then

$$\Pr(\mathcal{E} = e) = \binom{n^4}{e} \varepsilon^e (1-\varepsilon)^{n^4 - e}, \tag{19}$$

with $E[\mathcal{E}] = n^4 \varepsilon$.

Consider a Sudoku constrained array X. After transmission over the channel $X \to Y$ the received array Y may have some erased entries. In particular let X be the $(1, 1)$

array

1	2	3	4
3	4	1	2
2	1	4	3
4	3	2	1

after transmission through the channel a possible outcome for Y is

1	2	3	4
?	?	1	2
2	1	4	3
?	?	2	1

While this channel model is limited in its practical applications, it has been shown to provide a lot of insight into more complex channels, such as the binary symmetric channel or additive white Gaussian noise channel.

C. Decoding

A lot of work has gone into developing efficient solvers for Sudoku [8, 9, 10]. While most of this work does not consider possible communications applications, these solvers can just as easily function as decoders. Therefore, little attention is given to the decoding process, and a brute force maximum-likelihood decoder is used.

The decoder enumerates all feasible Sudoku constrained arrays. Let

$$I_e = \{i \in \{1, 2, \ldots, n^4\} \mid Y_i = ?\} \qquad I_r = \{i \in \{1, 2, \ldots, n^4\} \mid Y_i \neq ?\}, \tag{20}$$

be the ordered sets of erased and received entries where

$$Y(I_a) = \begin{bmatrix} Y_{i_1} & Y_{i_2} & \ldots & Y_{i_{|I_a|}} \end{bmatrix} \quad i_j \in I_a \text{ with } i_j < i_{j'} \text{ for } j < j', \tag{21}$$

is the corresponding erased or receive vector.

Definition: The decoding function $D : \mathcal{C} \to \mathcal{C}$ is

$$D(Y) := \{X \in S_n \mid X(I_r) = Y(I_r)\}. \tag{22}$$

That is, given a particular erasure pattern the decoder will determine all possible Sudoku constrained arrays that coincide with the non erased entries. For example consider the following received array

1	2	3	4
?	?	1	2
2	1	4	3
?	?	2	1

This array is not uniquely decodable since there are multiple combinations for the erased entries, in particular the following two arrays are valid.

1	2	3	4
3	4	1	2
2	1	4	3
4	3	2	1

1	2	3	4
4	3	1	2
2	1	4	3
3	4	2	1

D. Error patterns and simulations

Using the two arrays designated to represent the equivalence classes all 2^{16} possible erasure paterns are enumerated, the Matlab code is provided in Appendix B. Tables III and IV present the results of this enumeration. The primary concern for an effective channel coding scheme is unique decodability. Since the encoder and decoder operate within the collection of Sudoku constrained arrays, decoder failure will occur if there is more than one feasible Sudoku constrained array. That is, $|D(Y)| > 1$, or more explicitly if the erasure pattern is such that there are multiple valid symbols for a particular entry.

Table III. Symmetric Error Matrix

	Number of Erasures													
$\|D(Y)\|$	4	5	6	7	8	9	10	11	12	13	14	15	16	sum
2	8	96	528	1728	3650	5008	4200	1792	268					17278
3				16	144	576	1248	1424	608	48				4064
4					12	96	288	384	176					956
5							32	128	64					224
6							32	240	624	240				1136
7									12					12
8									16					16
9										48				48
10									16					16
12									20	208	16			244
18									4	16	32			52
24											56			56
36											16			16
72												16		16
288													1	1
sum	8	96	528	1744	3806	5680	5800	3968	1808	560	120	16	1	24135
total	1820	4368	8008	11440	12870	11440	8008	4368	1820	560	120	16	1	65536

Table IV. Asymmetric Error Matrix

	Number of Erasures													
$\|D(Y)\|$	4	5	6	7	8	9	10	11	12	13	14	15	16	sum
2	4	48	264	864	1830	2544	2240	1136	232					9162
3				16	144	576	1248	1424	608	48				4064
4					6	48	144	192	128					518
5							24	96	48					168
6							32	240	624	240				1136
7									8					8
8									8					8
9										48				48
10									12					12
12									20	208	16			244
18									4	16	32			52
24											56			56
36											16			16
72												16		16
288													1	1
sum	4	48	264	880	1980	3168	3688	3088	1692	560	120	16	1	15509
total	1820	4368	8008	11440	12870	11440	8008	4368	1820	560	120	16	1	65536

These tables provide a lot of insight into the error correcting capability of Sudoku constrained arrays. In particular, all errors less than or equal to 3 are correctable and no errors greater than 12 are correctable. Additionally, examination of the bottom row provides an explicit representation for the probability of failure as a function of epsilon. Let $\{e_i^{(j)} \mid i = 1, \ldots, 16 \; j = 1, 2\}$ denote the number of size i error patterns that result in error for symmetric and asymmetric arrays, 1 and 2 respectively. Then

the total probability of error as a function of epsilon is

$$P_e^j(\varepsilon) = \sum_{i=1}^{16} \frac{e_i^j}{\binom{16}{i}} \varepsilon^i (1-\varepsilon)^{16-i}. \tag{23}$$

This error probability has an intrinsic dependence on the symmetry of the array, denoted by j, and examining the error enumeration of these two types shows that the asymmetric arrays perform significantly better against erasures. Since there are 288 total elements in S_2, we can hope to encode at most $\lfloor \log 288 \rfloor = 8$ bits. Hence there are $288 - 256 = 32$ unused arrays, and therefore not all of the symmetric arrays need to be used.

Referencing Table II there are 192 asymmetric and 96 symmetric arrays, hence the total probability of error is

$$P_e(\varepsilon) = \sum_{i=1}^{16} \frac{1}{\binom{16}{i}} \left(\tfrac{64}{256} e_i^1 + \tfrac{192}{256} e_i^2 \right) \varepsilon^i (1-\varepsilon)^{16-i}. \tag{24}$$

Figure 5 plots this function for 1000 values of ε in $[\frac{1}{64}, \frac{1}{2}]$.

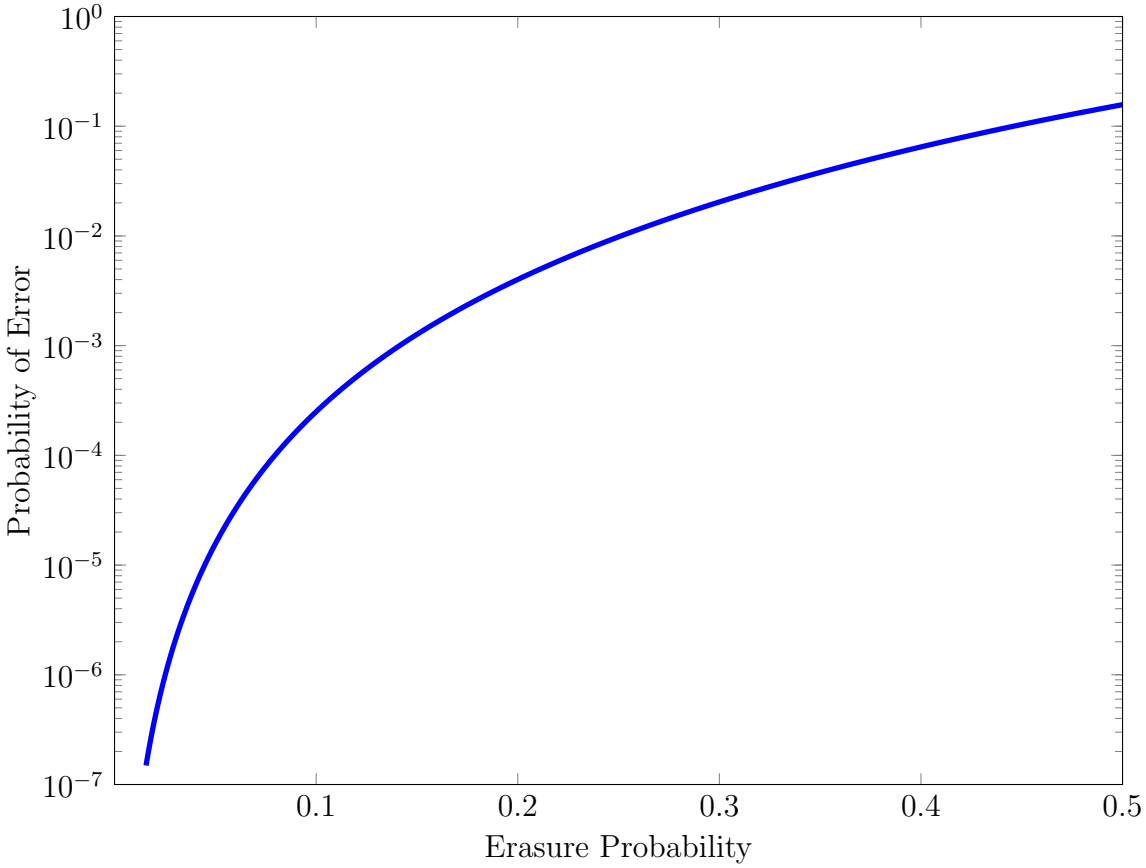


Fig. 5. Explicit Error Probability

This figure exhibits the error correcting capability of the the $n = 2$ Sudoku code. One major drawback of this encoding scheme is the low rate $\frac{1}{4}$. Part of this is the result of integer truncation, but in general the Sudoku constraints are very limiting. There is little practicality in an error correcting code based on Sudoku, and there are several codes with better performance. For example, a similar simulation was performed using a rate $\frac{1}{4}$ random linear code and their was a significant improvement in the probability of error.

CHAPTER V

PATHS

We now consider the problem of the optimal order to place entries into a Sudoku constrained array. That is, given we know an array is Sudoku constrained what is the minimal number of entries required to obtain complete knowledge of the array.

Consider the canonical labeling described in Section 1,

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_{n^2} \\ x_{n^2+1} & x_{n^2+2} & \dots & x_{n^4} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(n^2-1)n^2+1} & x_{(n^2-1)n^2+2} & \dots & x_{n^4} \end{bmatrix}. \tag{25}$$

Definition: A path, γ, is a permutation map $\gamma : (1, \dots, n^4) \to (\gamma_1, \dots, \gamma_{n^4})$

Definition: A fixed path, τ, is the concatenation of a path γ with an alphabet vector, $\alpha \in \mathcal{A}^{n^4}$, where $\mathcal{A} = \{1, \dots, n^2\}$, $\tau = (\gamma; \alpha)$.

Definition: A length k segmented fixed path is the restriction of a fixed path to the first k entries $\tau(k) = (\gamma_1, \dots, \gamma_k; \alpha_1, \dots, \alpha_k)$

Given a fixed path, entries are placed into an array using the following algorithm. An empty array is first initialized,

$$X_{\tau(0)} = \begin{bmatrix} \ \end{bmatrix}. \tag{26}$$

Entries are then placed into the array along γ by the following recursion

$$X_{\tau(k)}(\gamma(k+1)) = \alpha(k+1) \tag{27}$$

$$X_{\tau(k+1)} = X_{\tau(k)}, \tag{28}$$

the final step or $X_{\tau(n^4)}$ will simply be denoted X_τ.

Definition: A fixed path is valid if $X_\tau \in S_n$.

Definition: The kth order Sudoku constrained collection for a fixed path τ is

$$\mathcal{C}(X_{\tau(k)}) := \{X \in S_n \mid X = X_{\tau'} \text{ for some } \tau' = (\tau(k), \tau''(n-k)) \text{ valid}\}. \tag{29}$$

Definition: The length of a fixed path is the minimum number of steps to achieve a unique array

$$\ell(\tau) := \min\{k \in \mathbb{N} \mid |\mathcal{C}(X_{\tau(k)})| = 1\}. \tag{30}$$

The definition of length is extended to a general path in an expected sense.

Definition: The length of a path is the expected length of all valid fixed path

$$\hat{\ell}(\gamma) := E[\ell(\tau)] = \sum_{\tau \text{ valid}} \Pr(\tau)\ell(\tau). \tag{31}$$

All alphabet vectors are assumed equiprobable and thusly, this reduces to

$$\hat{\ell}(\gamma) = \frac{1}{|S_n|} \sum_{\tau \text{ valid}} \ell(\tau). \tag{32}$$

Definition: A path, γ, is optimal if $\hat{\ell}(\gamma) \leq \hat{\ell}(\gamma')$ for all other γ'.

Since each γ is a permutation mapping, the total number of possible paths grows very quickly with n. In fact the total number of possible paths for a given n is $n^4!$. For each of these possible paths there an are additional $|S_n|$ valid paths. Hence the total number of valid paths for a given n is $|S_n|n^4!$. Even for $n = 2$ this number is very large, $288 \cdot 16! \sim 10^{15}$. Figure 6 plots the empirical probability mass function (PMF) of the path length for $n = 2$ for 10^4 randomly drawn permutation paths, the Matlab code is provided in Appendix C.

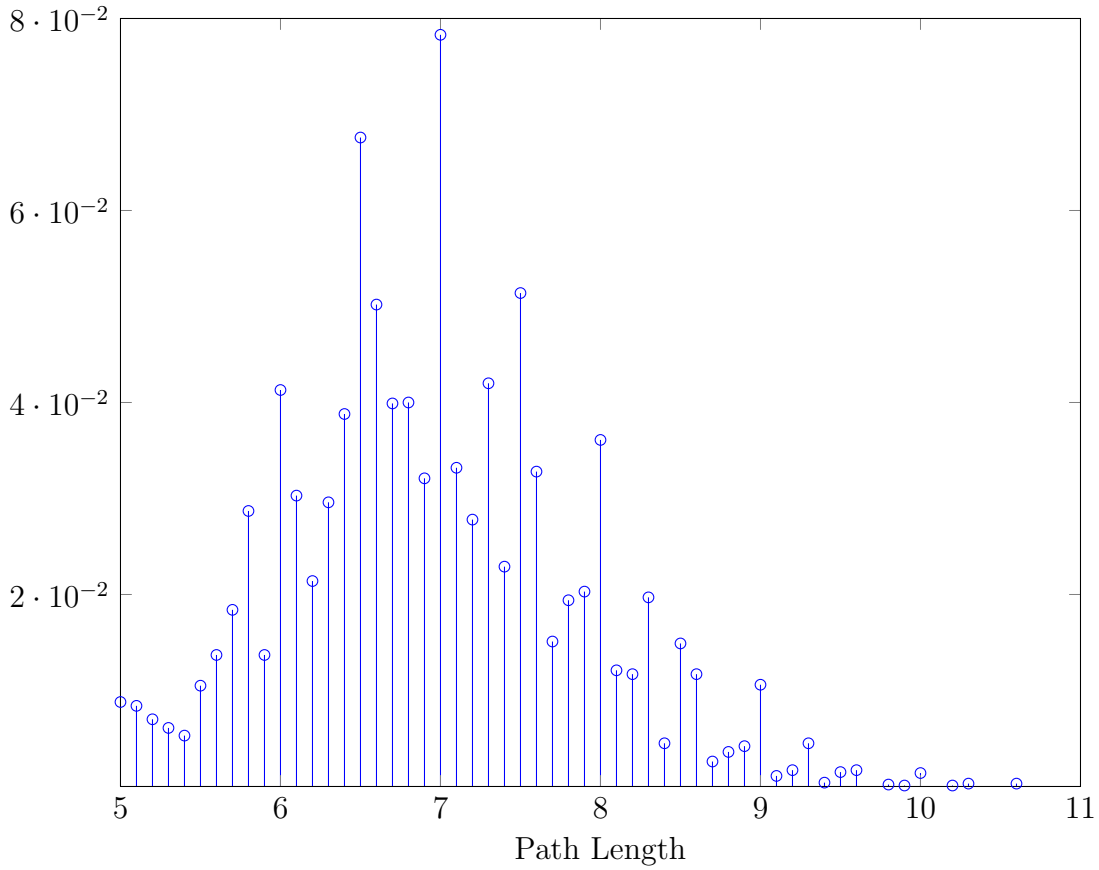
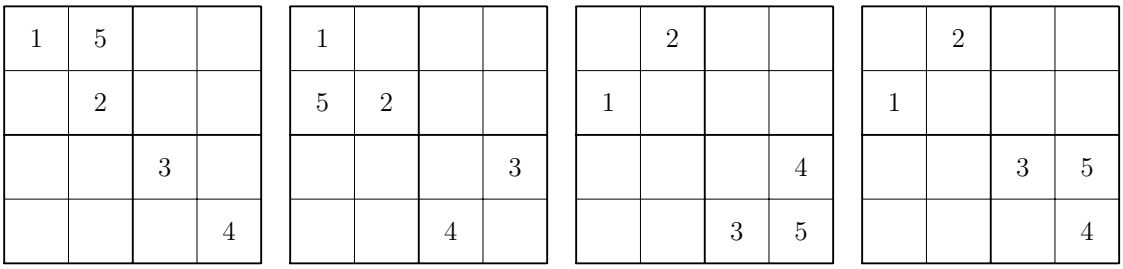


Fig. 6. Path Length Empirical PMF

The mean for this distribution is approximately 7 and the absolute minimum achievable length was found to be 5. This deviation shows that there is something to be gained by picking a nice path as opposed to choosing a path at random. A common theme of these optimal paths is sending diagonal entries. In particular, a path will have length 5 if the first four entries are sublock diagonals in two diagonal sublocks and the fifth is any other entry in one of those sublocks



These are just a few examples and there are many more possible combinations. In fact, if you take the first example of the main diagonal and apply any of the equivalence class operators discussed in section C it will maintain this property. An example of a length 5 path is provided in Figure 7, corresponding to

$$\gamma = (1, 11, 6, 16, 2, 12, 5, 15, 4, 10, 7, 13, 3, 9, 8, 14) \tag{33}$$

1	5	13	9
7	3	11	15
14	10	2	6
12	16	8	4

Fig. 7. An Optimal Path

The preceding analysis focused on the deterministic case, but a more important question is which path performs best under channel erasures. We now consider the case when erasures are possible, hence $\mathcal{A} = \{1, \ldots, n^2, ?\}$. This additional symbol provides an additional 2^{n^4} fixed paths, and thusly there are at most $|S_n|2^{n^4}n^4!$ possible valid paths.

One consideration of this extension is that the definition for path length requires a valid path, and since its unclear which of the new paths are valid, this definition

must be altered. One possibility is conditioning on the set of valid paths. However, this gives an unfair advantage to those paths with invalid fixed paths, since these paths are essentially weighted with zero.

Therefore, we shall use a different metric for path efficiancy.

Definition: The entropy of a length k segmented fixed path is

$$H(X_{\tau(k)}) := \log|D(X_{\tau(k)})|, \tag{34}$$

where D is the decoding function given in 22.

Let $\mathcal{A}_n(\gamma)$ denote the set of Sudoku constrained arrays, S_n, interleaved with all possible 2^{n^4} erasure patterns projected along γ. The probability of a particular element of this set, α, is a function of the number of erasures, $e(\alpha)$,

$$\Pr(\alpha) = \frac{1}{|S_n|}\varepsilon^{e(\alpha)}(1-\varepsilon)^{n^4-e(\alpha)}. \tag{35}$$

Definition: The entropy of a path is

$$H(\gamma) := \sum_{\alpha\in\mathcal{A}_n(\gamma)} \Pr(\alpha)\sum_{k=1}^{n^4} H(X_{\tau(k)}). \tag{36}$$

Definition: A path, γ, is optimal if $H(\gamma) \leq H(\gamma')$ for all other paths γ'.

CHAPTER VI

CONCLUSIONS

The Sudoku puzzle is a particular example in the broad class of constraint satisfaction problems. A mathematical formalization was provided to classify the set of Sudoku constrained arrays and, this collection was decomposed based on its intrinsic algebraic structure. Using this decomposition an explicit analysis was performed on the error correcting capability of Sudoku constrained arrays for the $n = 2$ case. The results of this analysis were less than ideal, as they provided little merit for the practical viability of an error correcting code using the Sudoku constraint. However, extending these ideas to a rateless framework and considering the optimal transmission path has provided some interesting results. Further analysis is required, but concatenation of the Sudoku constraint with rateless transmission may yield an effective error correcting code.

REFERENCES

[1] H. Lin and I. Wu, "Solving the minimum sudoku poblem," in *Proceedings of the 2010 International Conference on Technologies and Applications of Artificial Intelligence*, ser. TAAI '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 456–461. [Online]. Available: http://dx.doi.org/10.1109/TAAI.2010.77

[2] "History of sudoku," Oct. 2011, http://www.sudokudaily.net/history.php.

[3] J. Fraleigh and V. Katz, *A First Course in Abstract Algebra*, 7th ed. Boston, Massachusetts: Addison-Wesley, 2003.

[4] D. Bertsekas and J. Tsitsiklis, *Introduction to Probability*, 2nd ed. Belmont, Massachusetts: Athena Scientific, 2008.

[5] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, Massachusetts: Cambridge University Press, 2008.

[6] R. Gallager, *Principles of Digital Communication*. Cambridge, Massachusetts: Cambridge University Press, 2008.

[7] T. Cover and J. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, New Jersey: Wiley Interscience, 2006.

[8] T. Mantere and J. Koljonen, "Solving, rating and generating sudoku puzzles with ga," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, Sept. 2007, pp. 1382 –1389.

[9] T. Moon and J. Gunther, "Multiple constraint satisfaction by belief propagation: An example using sudoku," in *Adaptive and Learning Systems, 2006 IEEE Mountain Workshop on*, July 2006, pp. 122 –126.

[10] T. Moon, J. Gunther, and J. Kupin, "Sinkhorn solves sudoku," *Information Theory, IEEE Transactions on*, vol. 55, no. 4, pp. 1741 –1746, April 2009.

APPENDIX A

1.1 Symmetric group

1.1 Flip rotate subgroups

1.1 Flip rotate subgroup 1

1	2	3	4
3	4	1	2
2	1	4	3
4	3	2	1

4	2	3	1
3	1	4	2
2	4	1	3
1	3	2	4

4	3	2	1
2	1	4	3
3	4	1	2
1	2	3	4

1	3	2	4
2	4	1	3
3	1	4	2
4	2	3	1

Fig. 8. 1-1 1-22 1-24 1-3

1	2	4	3
4	3	1	2
2	1	3	4
3	4	2	1

3	2	4	1
4	1	3	2
2	3	1	4
1	4	2	3

3	4	2	1
2	1	3	4
4	3	1	2
1	2	4	3

1	4	2	3
2	3	1	4
4	1	3	2
3	2	4	1

Fig. 9. 1-2 1-16 1-18 1-5

1	3	4	2
4	2	1	3
3	1	2	4
2	4	3	1

2	3	4	1
4	1	2	3
3	2	1	4
1	4	3	2

2	4	3	1
3	1	2	4
4	2	1	3
1	3	4	2

1	4	3	2
3	2	1	4
4	1	2	3
2	3	4	1

Fig. 10. 1-4 1-10 1-12 1-6

2	1	3	4
3	4	2	1
1	2	4	3
4	3	1	2

4	1	3	2
3	2	4	1
1	4	2	3
2	3	1	4

4	3	1	2
1	2	4	3
3	4	2	1
2	1	3	4

2	3	1	4
1	4	2	3
3	2	4	1
4	1	3	2

Fig. 11. 1-7 1-20 1-23 1-9

2	1	4	3
4	3	2	1
1	2	3	4
3	4	1	2

3	1	4	2
4	2	3	1
1	3	2	4
2	4	1	3

3	4	1	2
1	2	3	4
4	3	2	1
2	1	4	3

2	4	1	3
1	3	2	4
4	2	3	1
3	1	4	2

Fig. 12. 1-8 1-14 1-17 1-11

3	1	2	4
2	4	3	1
1	3	4	2
4	2	1	3

4	1	2	3
2	3	4	1
1	4	3	2
3	2	1	4

4	2	1	3
3	1	4	2
2	4	3	1
3	1	2	4

3	2	1	4
1	4	3	2
2	3	4	1
4	1	2	3

Fig. 13. 1-13 1-19 1-20 1-15

1.2 Flip rotate subgroup 2

1	2	3	4
3	4	1	2
4	3	2	1
2	1	4	3

2	4	3	1
1	3	4	2
4	2	1	3
3	1	2	4

3	4	1	2
1	2	3	4
2	1	4	3
4	3	2	1

4	2	1	3
3	1	2	4
2	4	3	1
1	3	4	2

4	3	2	1
2	1	4	3
1	2	3	4
3	4	1	2

2	1	4	3
4	3	2	1
3	4	1	2
1	2	3	4

1	3	4	2
2	4	3	1
3	1	2	4
4	2	1	3

3	1	2	4
4	2	1	3
1	3	4	2
2	4	3	1

Fig. 14. 2-1 3-11 2-17 3-22 2-24 2-8 3-3 3-14

1	2	4	3
4	3	1	2
3	4	2	1
2	1	3	4

2	3	4	1
1	4	3	2
3	2	1	4
4	1	2	3

4	3	1	2
1	2	4	3
2	1	3	4
3	4	2	1

3	2	1	4
4	1	2	3
2	3	4	1
1	4	3	2

3	4	2	1
2	1	3	4
1	2	4	3
4	3	1	2

2	1	3	4
3	4	2	1
4	3	1	2
1	2	4	3

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

4	1	2	3
3	2	1	4
1	4	3	2
2	3	4	1

Fig. 15. 2-2 3-9 2-23 3-16 2-18 2-7 3-5 3-20

1	3	4	2
4	2	1	3
2	4	3	1
3	1	2	4

3	2	4	1
1	4	2	3
2	3	1	4
4	1	3	2

4	2	1	3
1	3	4	2
3	1	2	4
2	4	3	1

2	3	1	4
4	1	3	2
3	2	4	1
1	4	2	3

2	4	3	1
3	1	2	4
1	3	4	2
4	2	1	3

3	1	2	4
2	4	3	1
4	2	1	3
1	3	4	2

1	4	2	3
3	2	4	1
4	1	3	2
2	3	1	4

4	1	3	2
2	3	1	4
1	4	2	3
3	2	4	1

Fig. 16. 2-4 3-15 2-21 3-10 2-12 2-13 3-6 3-19

1	2	4	3
3	4	2	1
2	1	3	4
4	3	1	2

4	2	3	1
3	1	4	2
1	3	2	4
2	4	1	3

2	1	3	4
4	3	1	2
1	2	4	3
3	4	2	1

3	1	4	2
4	2	3	1
2	4	1	3
1	3	2	4

3	4	2	1
1	2	4	3
4	3	1	2
2	1	3	4

4	3	1	2
2	1	3	4
3	4	2	1
1	2	4	3

1	3	2	4
2	4	1	3
4	2	3	1
3	1	4	2

2	4	1	3
1	3	2	4
3	1	4	2
4	2	3	1

Fig. 17. 3-1 2-22 3-8 2-14 3-17 3-24 2-3 2-11

1	2	3	4
4	3	2	1
2	1	4	3
3	4	1	2

3	2	4	1
4	1	3	2
1	4	2	3
2	3	1	4

2	1	4	3
3	4	1	2
1	2	3	4
4	3	2	1

4	1	3	2
3	2	4	1
2	3	1	4
1	4	2	3

4	3	2	1
1	2	3	4
3	4	1	2
2	1	4	3

3	4	1	2
2	1	4	3
4	3	2	1
1	2	3	4

1	4	2	3
2	3	1	4
3	2	4	1
4	1	3	2

2	3	1	4
1	4	2	3
4	1	3	2
3	2	4	1

Fig. 18. 3-2 2-16 3-7 2-20 3-23 3-18 2-5 2-9

1	3	2	4
4	2	3	1
3	1	4	2
2	4	1	3

2	3	4	1
4	1	2	3
1	4	3	2
3	2	1	4

3	1	4	2
2	4	1	3
1	3	2	4
4	2	3	1

4	1	2	3
2	3	4	1
3	2	1	4
1	4	3	2

4	2	3	1
1	3	2	4
2	4	1	3
3	1	4	2

2	4	1	3
3	1	4	2
4	2	3	1
1	3	2	4

1	4	3	2
3	2	1	4
2	3	4	1
4	1	2	3

3	2	1	4
1	4	3	2
4	1	2	3
2	3	4	1

Fig. 19. 3-4 2-10 3-13 2-19 3-21 3-12 2-6 2-15

1.3 Flip rotate subgroup 3

1	2	4	3
3	4	2	1
4	3	1	2
2	1	3	4

2	4	3	1
1	3	4	2
3	1	2	4
4	2	1	3

4	3	1	2
2	1	3	4
1	2	4	3
3	4	2	1

3	1	2	4
4	2	1	3
2	4	3	1
1	3	4	2

3	4	2	1
1	2	4	3
2	1	3	4
4	3	1	2

2	1	3	4
4	3	1	2
3	4	2	1
1	2	4	3

1	3	4	2
2	4	3	1
4	2	1	3
3	1	2	4

4	2	1	3
3	1	2	4
1	3	4	2
2	4	3	1

Fig. 20. 4-1 4-11 4-24 4-14 4-17 4-8 4-3 4-22

1	2	3	4
4	3	2	1
3	4	1	2
2	1	4	3

2	3	4	1
1	4	3	2
4	1	2	3
3	2	1	4

3	4	1	2
2	1	4	3
1	2	3	4
4	3	2	1

4	1	2	3
3	2	1	4
2	3	4	1
1	4	3	2

4	3	2	1
1	2	3	4
2	1	4	3
3	4	1	2

2	1	4	3
3	4	1	2
4	3	2	1
1	2	3	4

1	4	3	2
2	3	4	1
3	2	1	4
4	1	2	3

3	2	1	4
4	1	2	3
1	4	3	2
2	3	4	1

Fig. 21. 4-2 4-9 4-18 4-20 4-23 4-7 4-5 4-16

1	3	2	4
4	2	3	1
2	4	1	3
3	1	4	2

3	2	4	1
1	4	2	3
4	1	3	2
2	3	1	4

2	4	1	3
3	1	4	2
1	3	2	4
4	2	3	1

4	1	3	2
2	3	1	4
3	2	4	1
1	4	2	3

4	2	3	1
1	3	2	4
3	1	4	2
2	4	1	3

3	1	4	2
2	4	1	3
4	2	3	1
1	3	2	4

1	4	2	3
3	2	4	1
2	3	1	4
4	1	3	2

2	3	1	4
4	1	3	2
1	4	2	3
3	2	4	1

Fig. 22. 4-4 4-15 4-12 4-19 4-21 4-13 4-6 4-10

1.2 Row column interchange subgroup

1	2	3	4
3	4	1	2
2	1	4	3
4	3	2	1

3	4	1	2
1	2	3	4
2	1	4	3
4	3	2	1

1	2	4	3
3	4	2	1
4	3	1	2
2	1	3	4

1	2	3	4
3	4	1	2
4	3	2	1
2	1	4	3

2	1	3	4
4	3	1	2
1	2	4	3
3	4	2	1

3	4	2	1
1	2	4	3
2	1	3	4
4	3	1	2

3	4	1	2
1	2	4	3
4	3	2	1
2	1	4	3

4	3	1	2
2	1	3	4
1	2	4	3
3	4	2	1

1	2	4	3
3	4	2	1
4	3	1	2
2	1	3	4

2	1	4	3
4	3	2	1
3	4	1	2
1	2	3	4

1	2	3	4
3	4	1	2
4	3	2	1
2	1	4	3

3	4	2	1
1	2	4	3
2	1	3	4
4	3	1	2

3	4	1	2
1	2	3	4
4	3	2	1
2	1	4	3

4	3	1	2
2	1	3	4
3	4	2	1
1	2	4	3

2	1	4	3
4	3	2	1
3	4	1	2
1	2	3	4

4	3	2	1
2	1	4	3
3	4	1	2
1	2	3	4

1	2	3	4
3	4	1	2
4	3	2	1
2	1	4	3

3	4	2	1
1	2	4	3
2	1	3	4
4	3	1	2

3	4	1	2
1	2	3	4
4	3	2	1
2	1	4	3

4	3	1	2
2	1	3	4
3	4	2	1
1	2	4	3

2	1	4	3
4	3	2	1
3	4	1	2
1	2	3	4

4	3	2	1
2	1	4	3
3	4	1	2
1	2	3	4

Fig. 23. 1-1 2-17 3-1 2-1 3-8 4-17 1-17 4-24 4-1 1-8 4-8 3-17 2-24 3-24 2-8 1-24

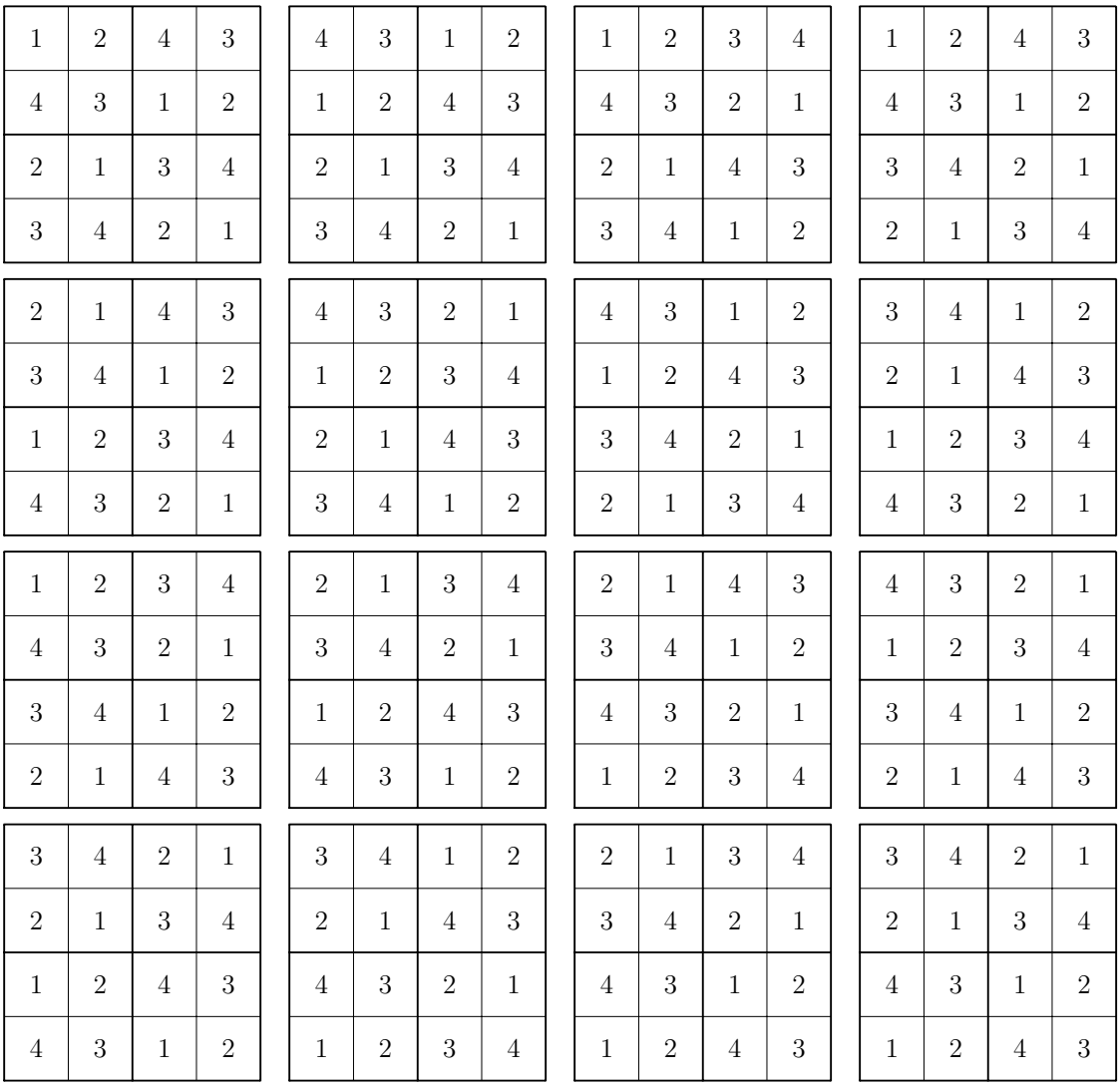


Fig. 24. 1-2 2-23 3-2 2-2 3-7 4-23 1-23 4-18 4-1 1-7 4-7 3-23 2-18 3-18 2-7 1-18

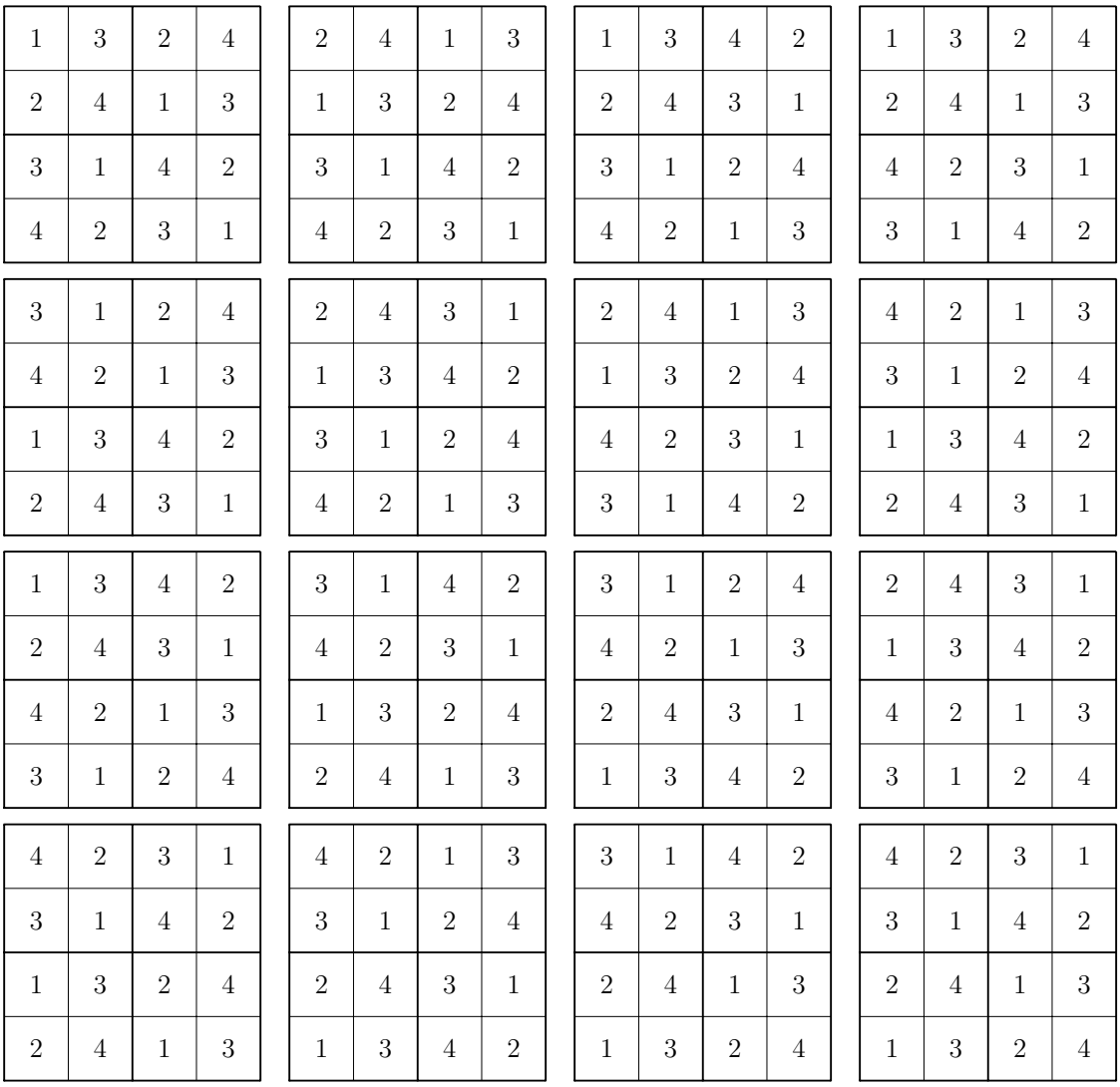


Fig. 25. 1-3 2-11 3-3 2-3 3-14 4-11 1-11 4-22 4-3 1-14 4-14 3-11 2-22 3-22 2-14 1-22

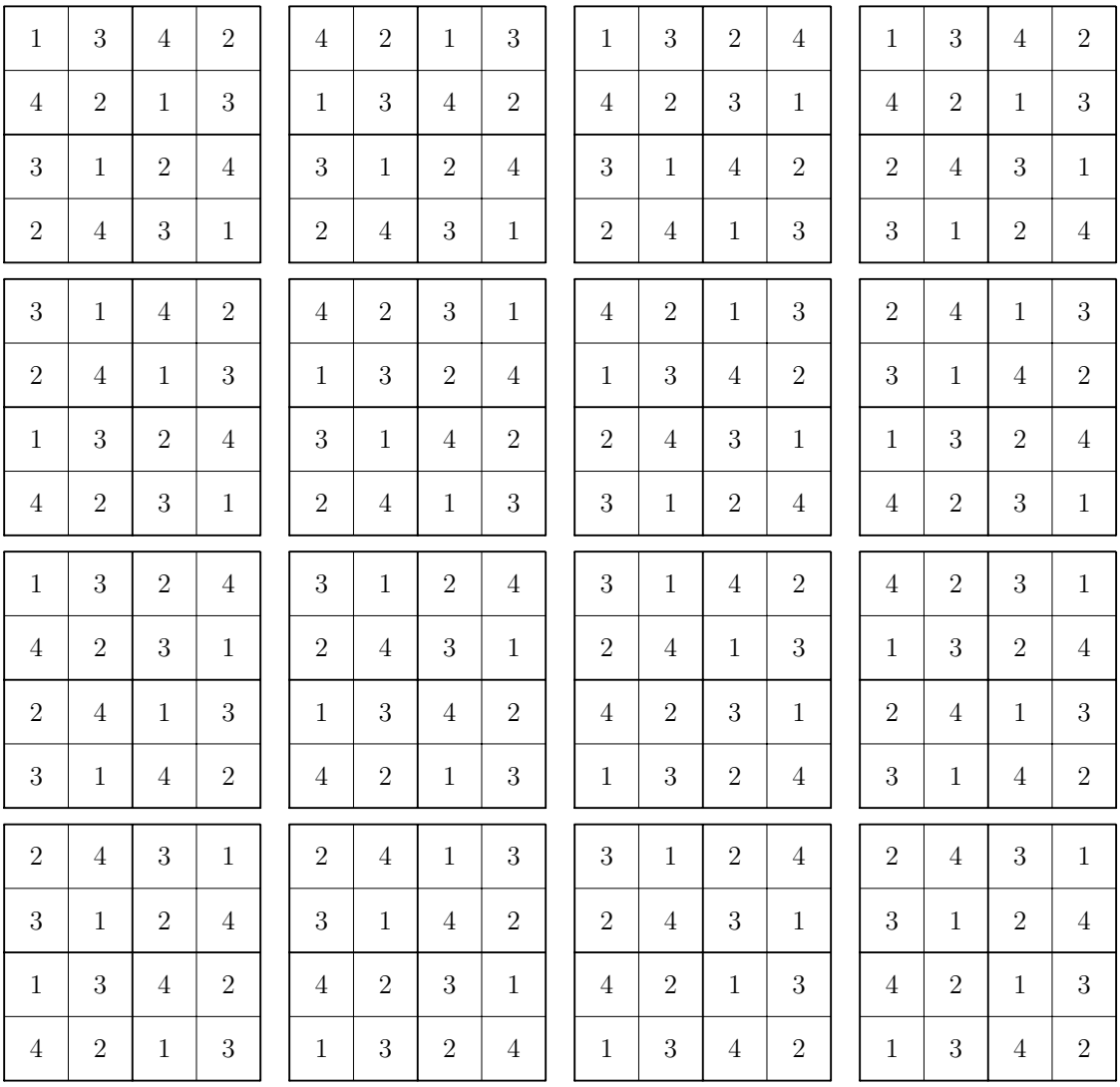


Fig. 26. 1-4 2-21 3-4 2-4 3-13 4-21 1-21 4-12 4-4 1-13 4-13 3-21 2-12 3-12 2-13 1-12

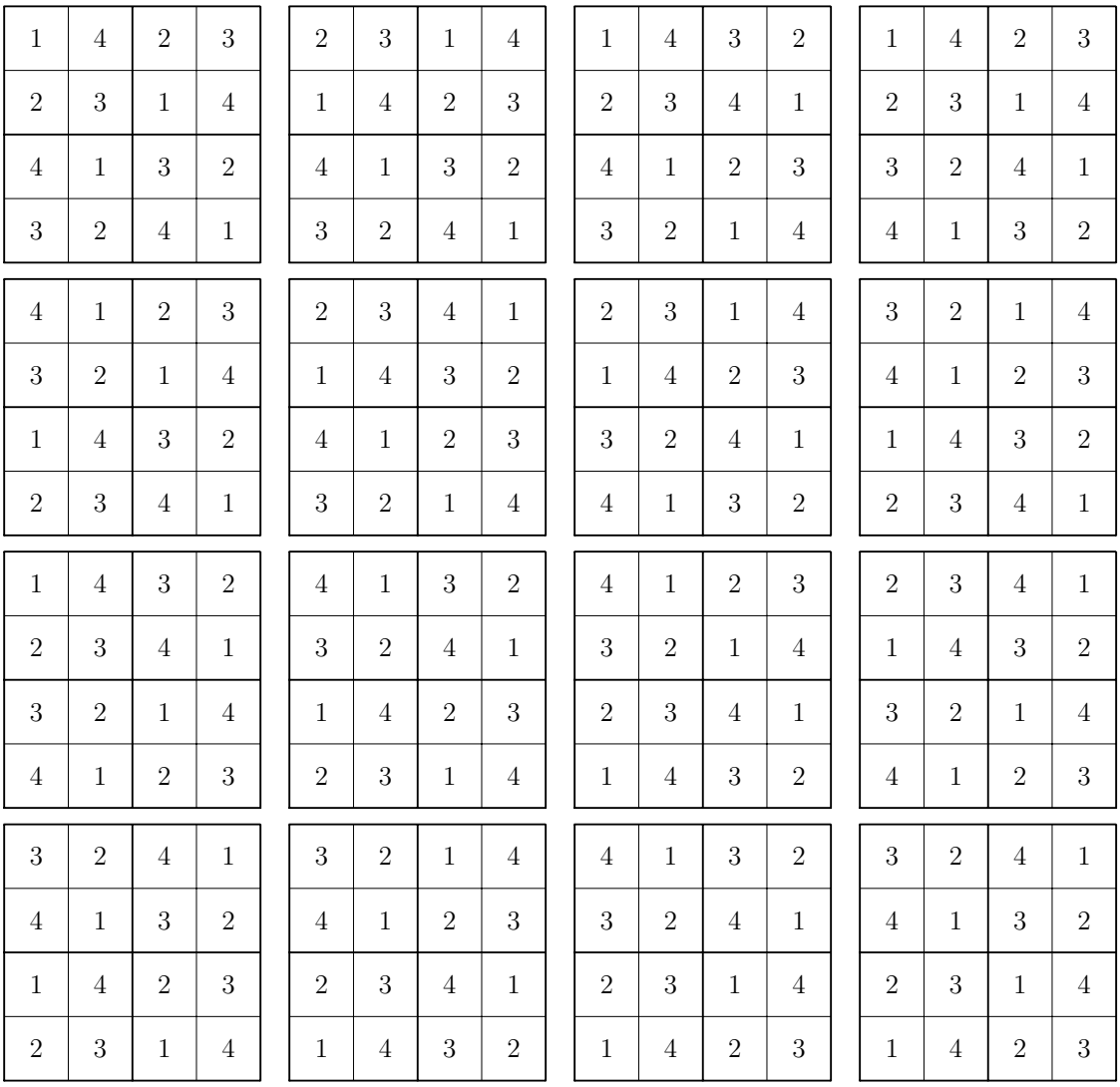


Fig. 27. 1-5 2-9 3-5 2-5 3-20 4-9 1-9 4-16 4-5 1-20 4-20 3-9 2-16 3-16 2-20 1-16

1	4	3	2
3	2	1	4
4	1	2	3
2	3	4	1

3	2	1	4
1	4	3	2
4	1	2	3
2	3	4	1

1	4	2	3
3	2	4	1
4	1	3	2
2	3	1	4

1	4	3	2
3	2	1	4
2	3	4	1
4	1	2	3

4	1	3	2
2	3	1	4
1	4	2	3
3	2	4	1

3	2	4	1
1	4	2	3
4	1	3	2
2	3	1	4

3	2	1	4
1	4	3	2
2	3	4	1
4	1	2	3

2	3	1	4
4	1	3	2
1	4	2	3
3	2	4	1

1	4	2	3
3	2	4	1
2	3	1	4
4	1	3	2

4	1	2	3
2	3	4	1
1	4	3	2
3	2	1	4

4	1	3	2
2	3	1	4
3	2	4	1
1	4	2	3

3	2	4	1
1	4	2	3
2	3	1	4
4	1	3	2

2	3	4	1
4	1	2	3
1	4	3	2
3	2	1	4

2	3	1	4
4	1	3	2
3	2	4	1
1	4	2	3

4	1	2	3
2	3	4	1
3	2	1	4
1	4	3	2

2	3	4	1
4	1	2	3
3	2	1	4
1	4	3	2

Fig. 28. 1-6 2-15 3-6 2-6 3-19 4-15 1-15 4-10 4-6 1-19 4-19 3-15 2-10 3-10 2-19 1-10

2.2 Asymmetric group

2.1 Flip rotate subgroups

2.1 Flip rotate subgroup 1

1	2	3	4
3	4	1	2
2	3	4	1
4	1	2	3

4	2	3	1
1	3	4	2
2	4	1	3
3	1	2	4

3	2	1	4
1	4	3	2
2	1	4	3
4	3	2	1

4	2	1	3
3	1	4	2
2	4	3	1
1	3	2	4

4	3	2	1
2	1	4	3
1	4	3	2
3	2	1	4

4	1	2	3
2	3	4	1
3	4	1	2
1	2	3	4

1	3	2	4
2	4	3	1
3	1	4	2
4	2	1	3

3	1	2	4
2	4	1	3
1	3	4	2
4	2	3	1

Fig. 29. 5-1 11-21 5-15 11-22 6-24 6-20 9-3 9-13

1	2	4	3
4	3	1	2
2	4	3	1
3	1	2	4

3	2	4	1
1	4	3	2
2	3	1	4
4	1	2	3

4	2	1	3
1	3	4	2
2	1	3	4
3	4	2	1

3	2	1	4
4	1	3	2
2	3	4	1
1	4	2	3

3	4	2	1
2	1	3	4
1	3	4	2
4	2	1	3

3	1	2	4
2	4	3	1
4	3	1	2
1	2	4	3

1	4	2	3
2	3	4	1
4	1	3	2
3	2	1	4

4	1	2	3
2	3	1	4
1	4	3	2
3	2	4	1

Fig. 30. 5-2 11-15 5-21 11-16 6-18 6-13 9-5 9-19

1	3	4	2
4	2	1	3
3	4	2	1
2	1	3	4

2	3	4	1
1	4	2	3
3	2	1	4
4	1	3	2

4	3	1	2
1	2	4	3
3	1	2	4
2	4	3	1

2	3	1	4
4	1	2	3
3	2	4	1
1	4	3	2

2	4	3	1
3	1	2	4
1	2	4	3
4	3	1	2

2	1	3	4
3	4	2	1
4	2	1	3
1	3	4	2

1	4	3	2
3	2	4	1
4	1	2	3
2	3	1	4

4	1	3	2
3	2	1	4
1	4	2	3
2	3	4	1

Fig. 31. 5-4 11-9 5-23 11-10 6-12 6-7 9-6 9-20

1	2	3	4
3	4	1	2
4	1	2	3
2	3	4	1

2	4	3	1
3	1	4	2
4	2	1	3
1	3	2	4

1	4	3	2
3	2	1	4
2	1	4	3
4	3	2	1

4	2	3	1
3	1	2	4
2	4	1	3
1	3	4	2

4	3	2	1
2	1	4	3
3	2	1	4
1	4	3	2

2	3	4	1
4	1	2	3
3	4	1	2
1	2	3	4

1	3	4	2
2	4	1	3
3	1	2	4
4	2	3	1

1	3	2	4
4	2	1	3
3	1	4	2
2	4	3	1

Fig. 32. 6-1 9-12 6-6 9-22 5-24 5-10 11-3 11-4

1	2	4	3
4	3	1	2
3	1	2	4
2	4	3	1

2	3	4	1
4	1	3	2
3	2	1	4
1	4	2	3

1	3	4	2
4	2	1	3
2	1	3	4
3	4	2	1

3	2	4	1
4	1	2	3
2	3	1	4
1	4	3	2

3	4	2	1
2	1	3	4
4	2	1	3
1	3	4	2

2	4	3	1
3	1	2	4
4	3	1	2
1	2	4	3

1	4	3	2
2	3	1	4
4	1	2	3
3	2	4	1

1	4	2	3
3	2	1	4
4	1	3	2
2	3	4	1

Fig. 33. 6-2 9-10 6-4 9-16 5-18 5-12 11-5 11-6

2	1	4	3
4	3	2	1
3	2	1	4
1	4	3	2

1	3	4	2
4	2	3	1
3	1	2	4
2	4	1	3

2	3	4	1
4	1	2	3
1	2	3	4
3	4	1	2

3	1	4	2
4	2	1	3
1	3	2	4
2	4	3	1

3	4	1	2
1	2	3	4
4	1	2	3
2	3	4	1

1	4	3	2
3	2	1	4
4	3	2	1
2	1	4	3

2	4	3	1
1	3	2	4
4	2	1	3
3	1	4	2

2	4	1	3
3	1	2	4
4	2	3	1
1	3	4	2

Fig. 34. 6-8 9-4 6-10 9-14 5-17 5-6 11-11 11-12

1	2	3	4
3	4	2	1
2	1	4	3
4	3	1	2

4	2	3	1
3	1	4	2
1	4	2	3
2	3	1	4

2	1	3	4
3	4	1	2
1	2	4	3
4	3	2	1

4	1	3	2
3	2	4	1
2	4	1	3
1	3	2	4

4	3	2	1
1	2	4	3
3	4	1	2
2	1	3	4

4	3	1	2
2	1	4	3
3	4	2	1
1	2	3	4

1	3	2	4
2	4	1	3
3	2	4	1
4	1	3	2

2	3	1	4
1	4	2	3
3	1	4	2
4	2	3	1

Fig. 35. 9-1 6-22 9-7 6-20 11-23 11-24 5-3 5-9

1	2	4	3
4	3	2	1
2	1	3	4
3	4	1	2

3	2	4	1
4	1	3	2
1	3	2	4
2	4	1	3

2	1	4	3
4	3	1	2
1	2	3	4
3	4	2	1

3	1	4	2
4	2	3	1
2	3	1	4
1	4	2	3

3	4	2	1
1	2	3	4
4	3	1	2
2	1	4	3

3	4	1	2
2	1	3	4
4	3	2	1
1	2	4	3

1	4	2	3
2	3	1	4
4	2	3	1
3	1	4	2

2	4	1	3
1	3	2	4
4	1	3	2
3	2	4	1

Fig. 36. 9-2 6-16 9-8 6-14 11-17 11-18 5-5 5-11

2	3	1	4
1	4	3	2
3	2	4	1
4	1	2	3

4	3	1	2
1	2	4	3
2	4	3	1
3	1	2	4

3	2	1	4
1	4	2	3
2	3	4	1
4	1	3	2

4	2	1	3
1	3	4	2
3	4	2	1
2	1	3	4

4	1	3	2
2	3	4	1
1	4	2	3
3	2	1	4

4	1	2	3
3	2	4	1
1	4	3	2
2	3	1	4

2	1	3	4
3	4	2	1
1	3	4	2
4	2	1	3

3	1	2	4
2	4	3	1
1	2	4	3
4	3	1	2

Fig. 37. 9-9 6-23 9-15 6-21 11-19 11-20 5-7 5-13

1	2	4	3
3	4	1	2
2	1	3	4
4	3	2	1

4	2	3	1
3	1	4	2
2	3	1	4
1	4	2	3

1	2	3	4
4	3	1	2
2	1	4	3
3	4	2	1

3	2	4	1
4	1	3	2
2	4	1	3
1	3	2	4

3	4	2	1
2	1	4	3
4	3	1	2
1	2	3	4

4	3	2	1
2	1	3	4
3	4	1	2
1	2	4	3

1	3	2	4
2	4	1	3
4	1	3	2
3	2	4	1

1	4	2	3
2	3	1	4
3	1	4	2
4	2	3	1

Fig. 38. 11-1 5-22 11-2 5-16 9-18 9-24 6-3 6-5

2	1	4	3
3	4	2	1
1	2	3	4
4	3	1	2

4	1	3	2
3	2	4	1
1	3	2	4
2	4	1	3

2	1	3	4
4	3	2	1
1	2	4	3
3	4	1	2

3	1	4	2
4	2	3	1
1	4	2	3
2	3	1	4

3	4	1	2
1	2	4	3
4	3	2	1
2	1	3	4

4	3	1	2
1	2	3	4
3	4	2	1
2	1	4	3

2	3	1	4
1	4	2	3
4	2	3	1
3	1	4	2

2	4	1	3
1	3	2	4
3	2	4	1
4	1	3	2

Fig. 39. 11-7 5-20 11-8 5-14 9-17 9-23 6-9 6-11

3	1	4	2
2	4	3	1
1	3	2	4
4	2	1	3

4	1	2	3
2	3	4	1
1	2	3	4
3	4	1	2

3	1	2	4
4	2	3	1
1	3	4	2
2	4	1	3

2	1	4	3
4	3	2	1
1	4	3	2
3	2	1	4

2	4	1	3
1	3	4	2
4	2	3	1
3	1	2	4

4	2	1	3
1	3	2	4
2	4	3	1
3	1	4	2

3	2	1	4
1	4	3	2
4	3	2	1
2	1	4	3

3	4	1	2
1	2	3	4
2	3	4	1
4	1	2	3

Fig. 40. 11-13 5-19 11-14 5-8 9-11 9-21 6-15 6-17

2.2 Flip rotate subgroup 2

1	2	4	3
3	4	2	1
2	3	1	4
4	1	3	2

4	2	3	1
1	3	4	2
3	1	2	4
2	4	1	3

2	3	1	4
4	1	3	2
1	2	4	3
3	4	2	1

3	1	4	2
4	2	1	3
2	4	3	1
1	3	2	4

3	4	2	1
1	2	4	3
4	1	3	2
2	3	1	4

4	1	3	2
2	3	1	4
3	4	2	1
1	2	4	3

1	3	2	4
2	4	3	1
4	2	1	3
3	1	4	2

2	4	1	3
3	1	2	4
1	3	4	2
4	2	3	1

Fig. 41. 7-1 12-21 8-10 10-13 7-17 8-19 10-3 12-12

1	2	3	4
4	3	2	1
2	4	1	3
3	1	4	2

3	2	4	1
1	4	3	2
4	1	2	3
2	3	1	4

2	4	1	3
3	1	4	2
1	2	3	4
4	3	2	1

4	1	3	2
3	2	1	4
2	3	4	1
1	4	2	3

4	3	2	1
1	2	3	4
3	1	4	2
2	4	1	3

3	1	4	2
2	4	1	3
4	3	2	1
1	2	3	4

1	4	2	3
2	3	4	1
3	2	1	4
4	1	3	2

2	3	1	4
4	1	2	3
1	4	3	2
3	2	4	1

Fig. 42. 7-2 12-15 8-12 10-20 7-23 8-13 10-5 12-10

1	3	2	4
4	2	3	1
3	4	1	2
2	1	4	3

2	3	4	1
1	4	2	3
4	1	3	2
3	2	1	4

3	4	1	2
2	1	4	3
1	3	2	4
4	2	3	1

4	1	2	3
2	3	1	4
3	2	4	1
1	4	3	2

4	2	3	1
1	3	2	4
2	1	4	3
3	4	1	2

2	1	4	3
3	4	1	2
4	2	3	1
1	3	2	4

1	4	3	2
3	2	4	1
2	3	1	4
4	1	2	3

3	2	1	4
4	1	3	2
1	4	2	3
2	3	4	1

Fig. 43. 7-4 12-9 8-18 10-19 7-21 8-7 10-6 12-16

1	2	4	3
3	4	2	1
4	1	3	2
2	3	1	4

2	4	3	1
3	1	4	2
1	3	2	4
4	2	1	3

4	1	3	2
2	3	1	4
1	2	4	3
3	4	2	1

3	1	2	4
4	2	3	1
2	4	1	3
1	3	4	2

3	4	2	1
1	2	4	3
2	3	1	4
4	1	3	2

2	3	1	4
4	1	3	2
3	4	2	1
1	2	4	3

1	3	4	2
2	4	1	3
4	2	3	1
3	1	2	4

4	2	1	3
1	3	2	4
3	1	4	2
2	4	3	1

Fig. 44. 8-1 10-12 7-19 12-14 8-17 7-10 12-4 10-21

1	2	3	4
4	3	2	1
3	1	4	2
2	4	1	3

2	3	4	1
4	1	3	2
1	4	2	3
3	2	1	4

3	1	4	2
2	4	1	3
1	2	3	4
4	3	2	1

4	1	2	3
3	2	4	1
2	3	1	4
1	4	3	2

4	3	2	1
1	2	3	4
2	4	1	3
3	1	4	2

2	4	1	3
3	1	4	2
4	3	2	1
1	2	3	4

1	4	3	2
2	3	1	4
3	2	4	1
4	1	2	3

3	2	1	4
1	4	2	3
4	1	3	2
2	3	4	1

Fig. 45. 8-2 10-10 7-13 12-20 8-23 7-12 12-5 10-15

1	3	2	4
4	2	3	1
2	1	4	3
3	4	1	2

3	2	4	1
4	1	2	3
1	4	3	2
2	3	1	4

2	1	4	3
3	4	1	2
1	3	2	4
4	2	3	1

4	1	3	2
2	3	4	1
3	2	1	4
1	4	2	3

4	2	3	1
1	3	2	4
3	4	1	2
2	1	4	3

3	4	1	2
2	1	4	3
4	2	3	1
1	3	2	4

1	4	2	3
3	2	1	4
2	3	4	1
4	1	3	2

2	3	1	4
1	4	3	2
4	1	2	3
3	2	4	1

Fig. 46. 8-4 10-16 7-7 12-19 8-21 7-18 12-6 10-9

1	2	3	4
3	4	2	1
4	3	1	2
2	1	4	3

2	4	3	1
1	3	4	2
4	1	2	3
3	2	1	4

3	4	1	2
2	1	3	4
1	2	4	3
4	3	2	1

4	1	2	3
3	2	1	4
2	4	3	1
1	3	4	2

4	3	2	1
1	2	4	3
2	1	3	4
3	4	1	2

2	1	4	3
4	3	1	2
3	4	2	1
1	2	3	4

1	3	4	2
2	4	3	1
3	2	1	4
4	1	2	3

3	2	1	4
4	1	2	3
1	3	4	2
2	4	3	1

Fig. 47. 10-1 7-11 12-18 8-20 12-23 10-8 7-3 8-16

1	2	4	3
4	3	2	1
3	4	1	2
2	1	3	4

2	3	4	1
1	4	3	2
3	1	2	4
4	2	1	3

4	3	1	2
2	1	4	3
1	2	3	4
3	4	2	1

3	1	2	4
4	2	1	3
2	3	4	1
1	4	3	2

3	4	2	1
1	2	3	4
2	1	4	3
4	3	1	2

2	1	3	4
3	4	1	2
4	3	2	1
1	2	4	3

1	4	3	2
2	3	4	1
4	2	1	3
3	1	2	4

4	2	1	3
3	1	2	4
1	4	3	2
2	3	4	1

Fig. 48. 10-2 7-9 12-24 8-14 12-17 10-7 7-5 8-22

1	3	4	2
4	2	3	1
2	4	1	3
3	1	2	4

3	2	4	1
1	4	2	3
2	1	3	4
4	3	1	2

4	2	1	3
3	1	4	2
1	3	2	4
2	4	3	1

2	1	3	4
4	3	1	2
3	2	4	1
1	4	2	3

2	4	3	1
1	3	2	4
3	1	4	2
4	2	1	3

3	1	2	4
2	4	1	3
4	2	3	1
1	3	4	2

1	4	2	3
3	2	4	1
4	3	1	2
2	1	3	4

4	3	1	2
2	1	3	4
1	4	2	3
3	2	4	1

Fig. 49. 10-4 7-15 12-22 8-8 12-11 10-13 7-6 8-24

1	2	4	3
3	4	1	2
4	3	2	1
2	1	3	4

2	4	3	1
1	3	4	2
3	2	1	4
4	1	2	3

4	3	1	2
1	2	3	4
2	1	4	3
3	4	2	1

3	2	1	4
4	1	2	3
2	4	3	1
1	3	4	2

3	4	2	1
2	1	4	3
1	2	3	4
4	3	1	2

2	1	3	4
4	3	2	1
3	4	1	2
1	2	4	3

1	3	4	2
2	4	3	1
4	1	2	3
3	2	1	4

4	1	2	3
3	2	1	4
1	3	4	2
2	4	3	1

Fig. 50. 12-1 8-11 10-23 7-16 10-18 12-8 8-3 7-20

1	2	3	4
4	3	1	2
3	4	2	1
2	1	4	3

2	3	4	1
1	4	3	2
4	2	1	3
3	1	2	4

3	4	1	2
1	2	4	3
2	1	3	4
4	3	2	1

4	2	1	3
3	1	2	4
2	3	4	1
1	4	3	2

4	3	2	1
2	1	3	4
1	2	4	3
3	4	1	2

2	1	4	3
3	4	2	1
4	3	1	2
1	2	3	4

1	4	3	2
2	3	4	1
3	1	2	4
4	2	1	3

3	1	2	4
4	2	1	3
1	4	3	2
2	3	4	1

Fig. 51. 12-2 8-9 10-17 7-22 10-24 12-7 8-5 7-14

1	3	2	4
4	2	1	3
2	4	3	1
3	1	4	2

3	2	4	1
1	4	2	3
4	3	1	2
2	1	3	4

2	4	1	3
1	3	4	2
3	1	2	4
4	2	3	1

4	3	1	2
2	1	3	4
3	2	4	1
1	4	2	3

4	2	3	1
3	1	2	4
1	3	4	2
2	4	1	3

3	1	4	2
2	4	3	1
4	2	1	3
1	3	2	4

1	4	2	3
3	2	4	1
2	1	3	4
4	3	1	2

2	1	3	4
4	3	1	2
1	4	2	3
3	2	4	1

Fig. 52. 12-4 8-15 10-11 7-24 10-22 12-13 8-6 7-8

2.2 Row column interchange subgroups

2.1 Row column interchange subgroup 1

1	2	3	4
3	4	1	2
2	3	4	1
4	1	2	3

3	4	1	2
1	2	3	4
2	3	4	1
4	1	2	3

1	2	4	3
3	4	2	1
2	3	1	4
4	1	3	2

1	2	3	4
3	4	1	2
4	1	2	3
2	3	4	1

2	1	3	4
4	3	1	2
3	2	4	1
1	4	2	3

3	4	2	1
1	2	4	3
2	3	1	4
4	1	3	2

3	4	1	2
1	2	3	4
4	1	2	3
2	3	4	1

4	3	1	2
2	1	3	4
1	4	2	3
3	2	4	1

2	1	4	3
4	3	2	1
1	2	3	4
3	4	1	2

2	1	4	3
4	3	2	1
1	4	2	3
3	2	4	1

2	1	3	4
4	3	1	2
1	4	2	3
3	2	4	1

3	4	1	2
1	2	3	4
4	1	2	3
2	3	4	1

4	3	1	2
2	1	3	4
1	4	2	3
3	2	4	1

2	1	4	3
4	3	2	1
1	4	3	2
3	2	1	4

4	3	2	1
2	1	4	3
3	2	1	4
1	4	3	2

4	3	1	2
2	1	3	4
1	4	2	3
3	2	4	1

2	1	4	3
4	3	2	1
1	4	3	2
3	2	1	4

4	3	2	1
2	1	4	3
1	4	3	2
3	2	1	4

Fig. 53. 5-1 6-17 7-1 6-1 8-8 8-17 5-17 7-24 8-1 6-8 7-8 7-17 5-24 8-24 5-8 6-24

1	2	4	3
4	3	1	2
2	4	3	1
3	1	2	4

4	3	1	2
1	2	4	3
2	4	3	1
3	1	2	4

1	2	3	4
4	3	2	1
2	4	1	3
3	1	4	2

1	2	4	3
4	3	1	2
3	1	4	2
2	4	3	1

2	1	4	3
3	4	1	2
4	2	3	1
1	3	2	4

4	3	2	1
1	2	3	4
2	4	1	3
3	1	4	2

4	3	1	2
1	2	4	3
3	1	4	2
2	4	3	1

3	4	1	2
2	1	4	3
4	2	3	1
1	3	2	4

1	2	3	4
4	3	2	1
3	1	4	2
2	4	1	3

2	1	3	4
3	4	2	1
4	2	1	3
1	3	4	2

2	1	4	3
3	4	1	2
1	3	2	4
4	2	3	1

4	3	2	1
1	2	3	4
3	1	4	2
2	4	1	3

3	4	2	1
2	1	3	4
4	2	1	3
1	3	4	2

3	4	1	2
2	1	4	3
1	3	2	4
4	2	3	1

2	1	3	4
3	4	2	1
1	3	4	2
4	2	1	3

3	4	2	1
2	1	3	4
1	3	4	2
4	2	1	3

Fig. 54. 5-2 6-23 7-2 6-2 8-7 8-23 5-23 7-18 8-2 6-7 7-7 7-23 5-18 8-18 5-7 6-18

1	3	2	4
2	4	1	3
3	2	4	1
4	1	3	2

2	4	1	3
1	3	2	4
3	2	4	1
4	1	3	2

1	3	4	2
2	4	3	1
3	2	1	4
4	1	2	3

1	3	2	4
2	4	1	3
4	1	3	2
3	2	4	1

3	1	2	4
4	2	1	3
2	3	4	1
1	4	3	2

2	4	3	1
1	3	4	2
3	2	1	4
4	1	2	3

2	4	1	3
1	3	2	4
4	1	3	2
3	2	4	1

4	2	1	3
3	1	2	4
2	3	4	1
1	4	3	2

1	3	4	2
2	4	3	1
4	1	2	3
3	2	1	4

3	1	4	2
4	2	3	1
2	3	1	4
1	4	2	3

3	1	2	4
4	2	1	3
1	3	4	2
2	4	3	1

2	4	3	1
1	3	4	2
4	1	2	3
3	2	1	4

4	2	3	1
3	1	4	2
2	3	1	4
1	4	2	3

4	2	1	3
3	1	2	4
1	4	3	2
2	3	4	1

3	1	4	2
4	2	3	1
1	4	2	3
2	3	1	4

4	2	3	1
3	1	4	2
1	4	2	3
2	3	1	4

Fig. 55. 5-3 6-11 7-3 6-3 8-14 8-11 5-11 7-22 8-3 6-14 7-14 7-11 5-22 8-22 5-14 6-22

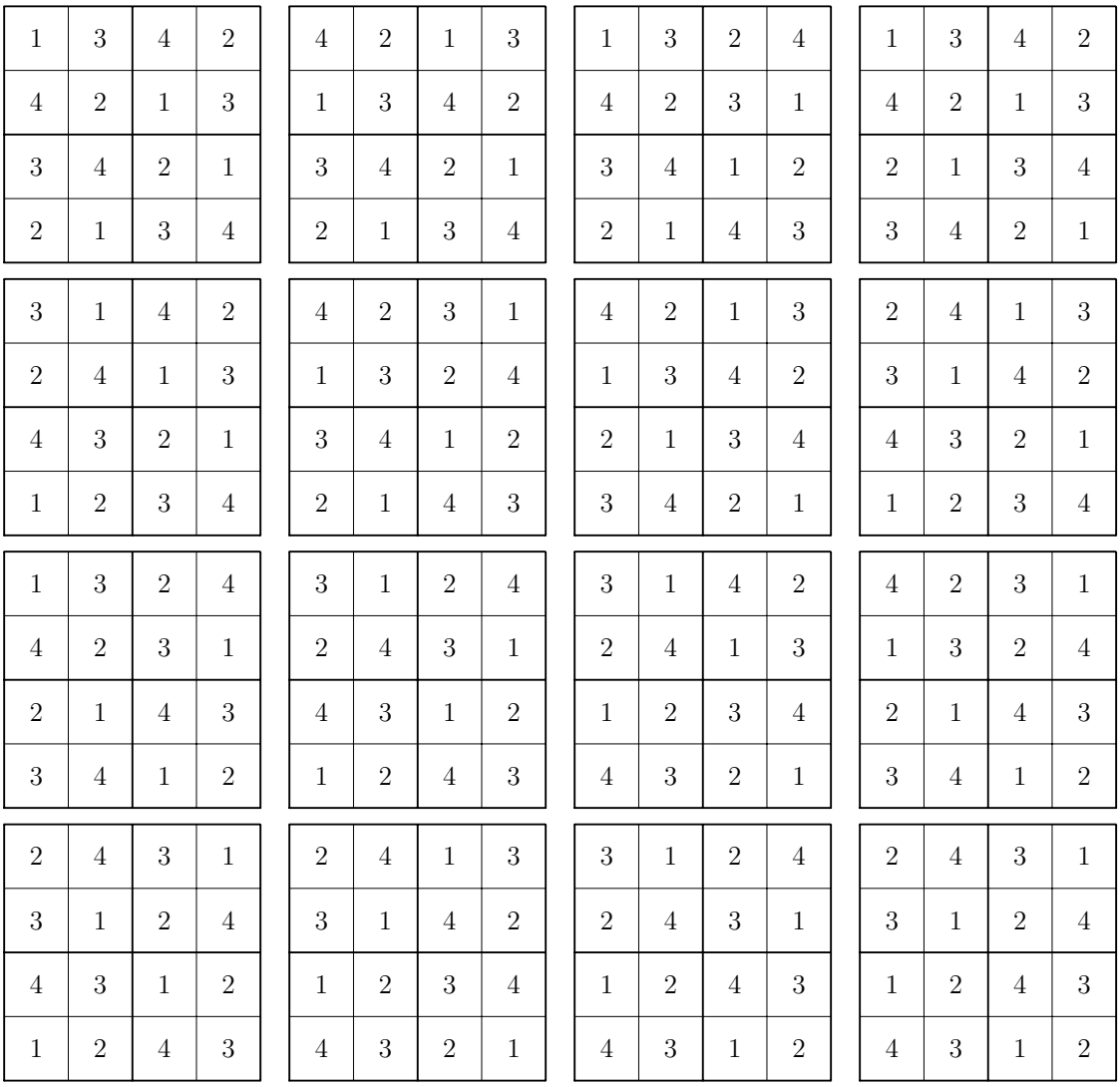


Fig. 56. 5-4 6-21 7-4 6-4 8-13 8-21 5-21 7-12 8-4 6-13 7-13 7-21 5-12 8-12 5-13 6-12

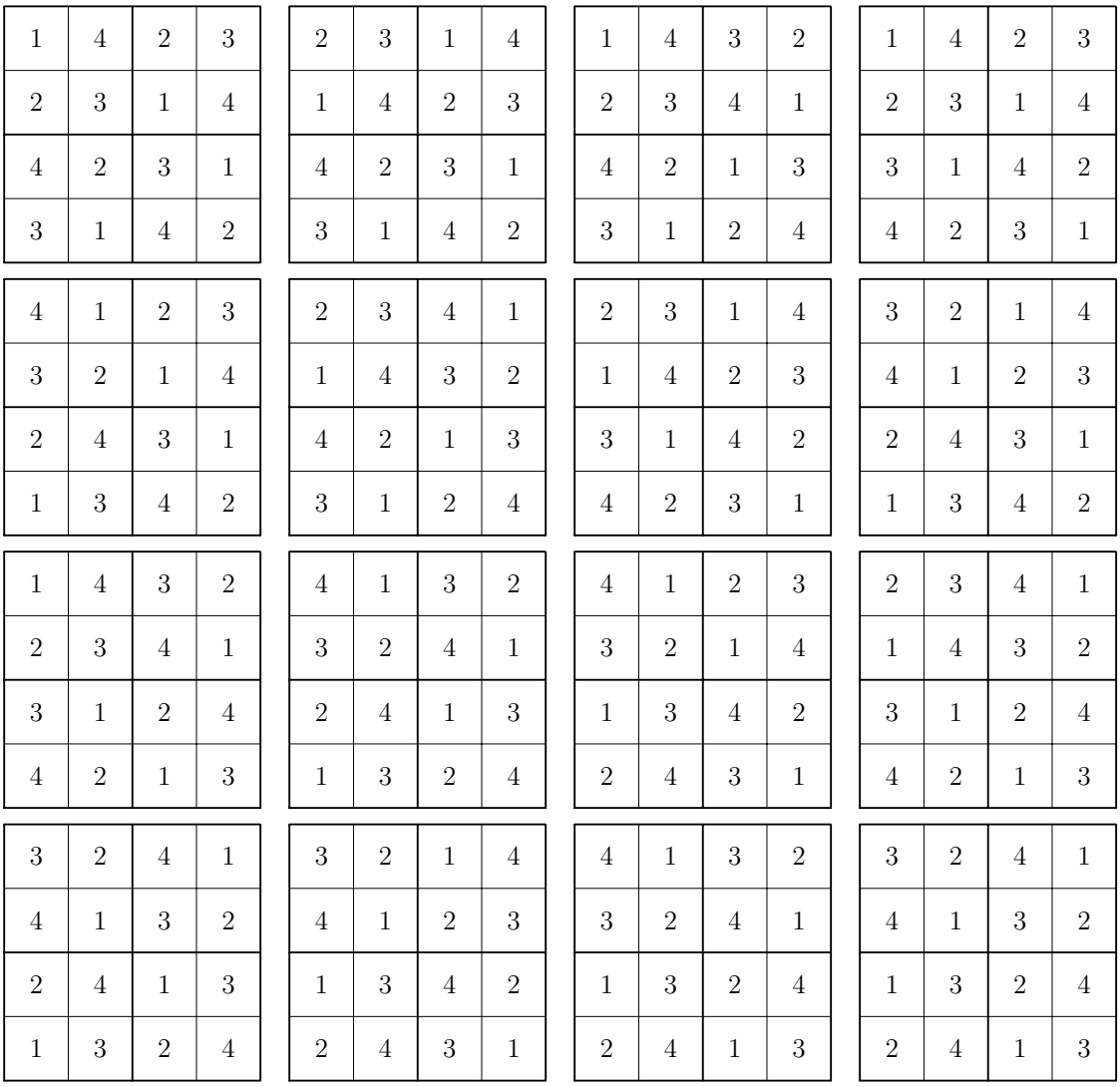


Fig. 57. 5-5 6-9 7-5 6-5 8-20 8-9 5-9 7-16 8-5 6-20 7-20 7-9 5-16 8-16 5-20 6-16

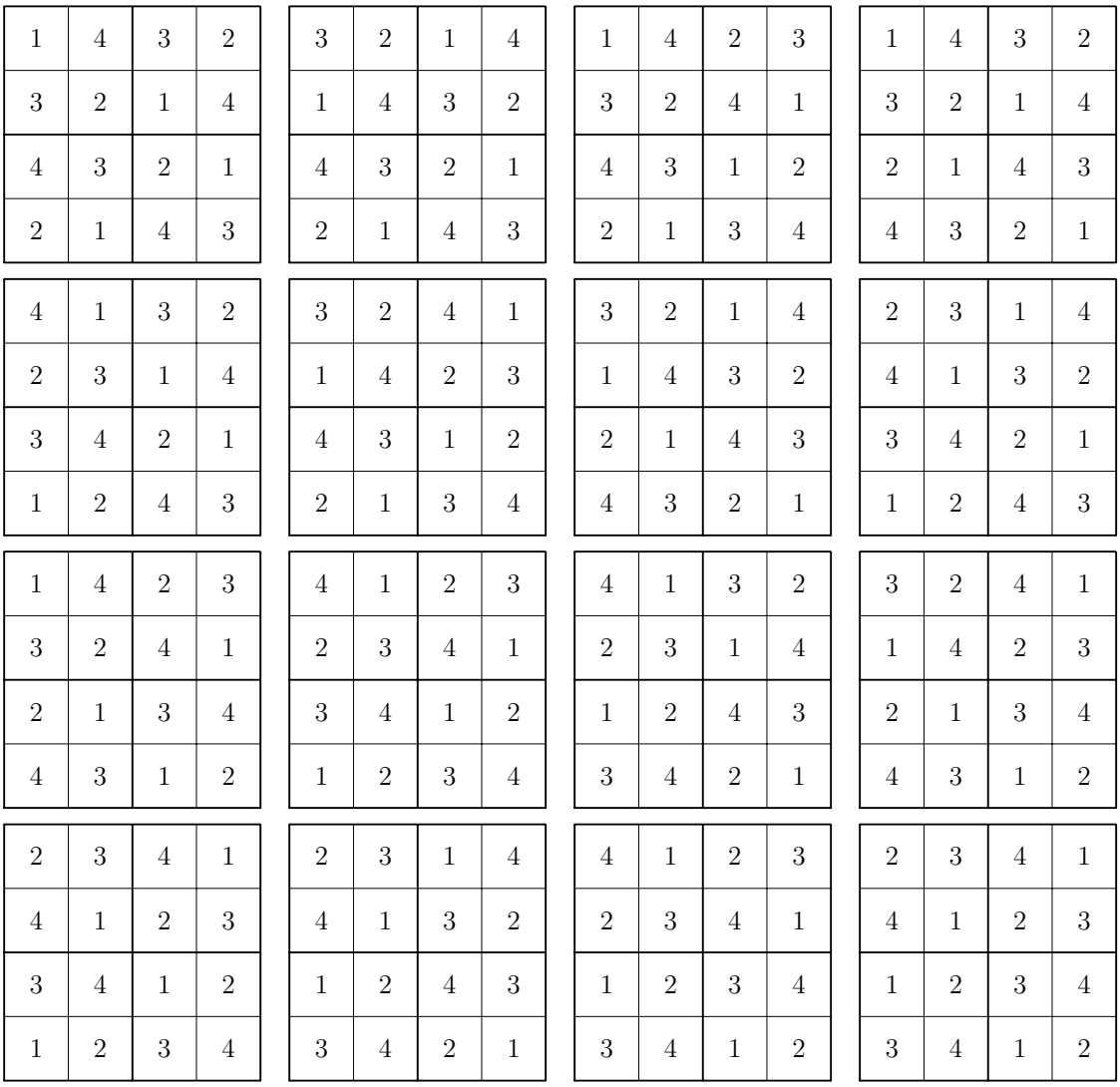


Fig. 58. 5-6 6-15 7-6 6-6 8-19 8-15 5-15 7-10 8-6 6-19 7-19 7-15 5-10 8-10 5-19 6-10

2.2 Row column interchange subgroup 2

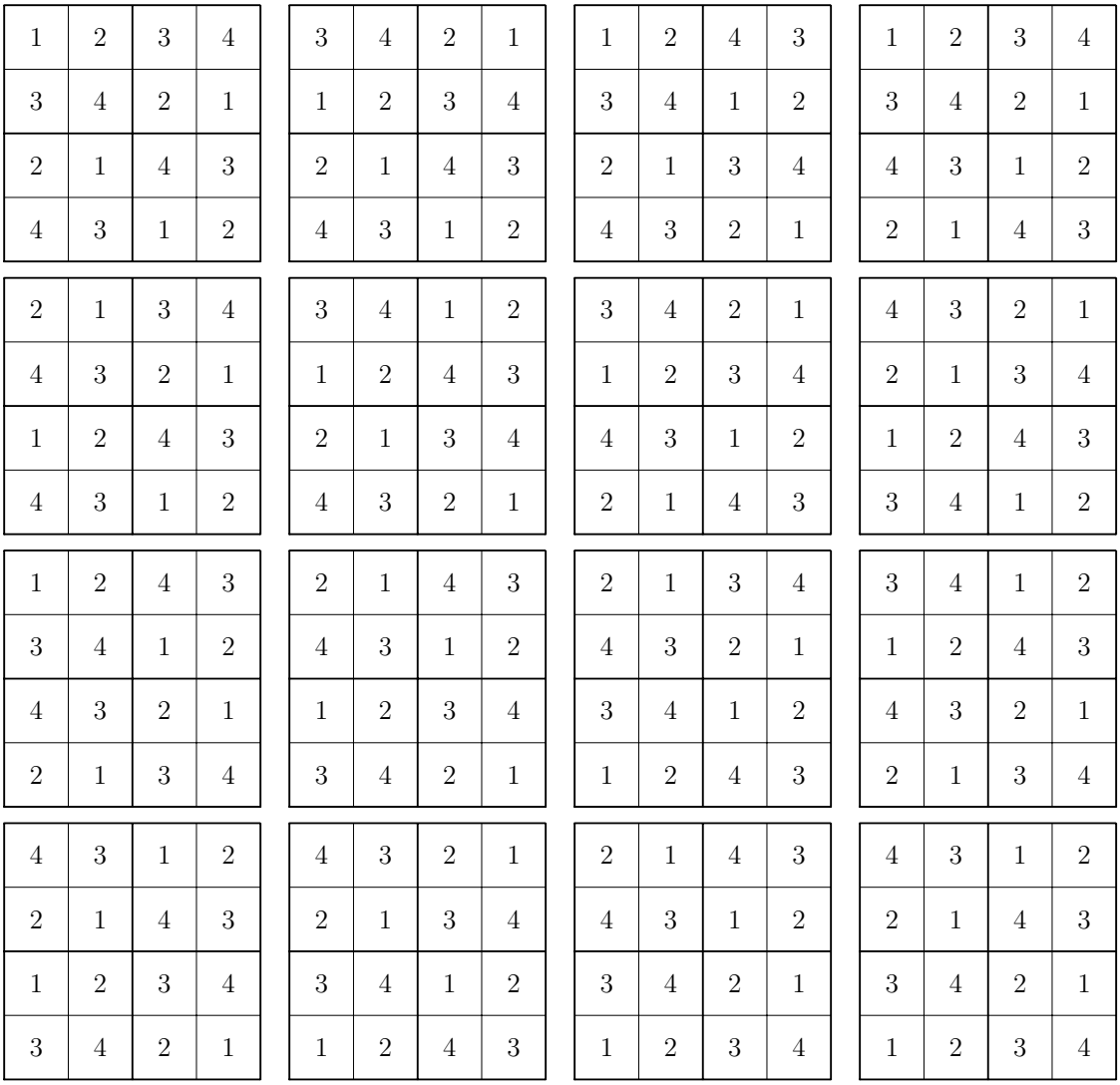


Fig. 59. 9-1 12-17 11-1 10-1 9-8 10-17 11-17 10-24 12-1 10-8 12-8 9-17 12-24 9-24 11-8 11-24

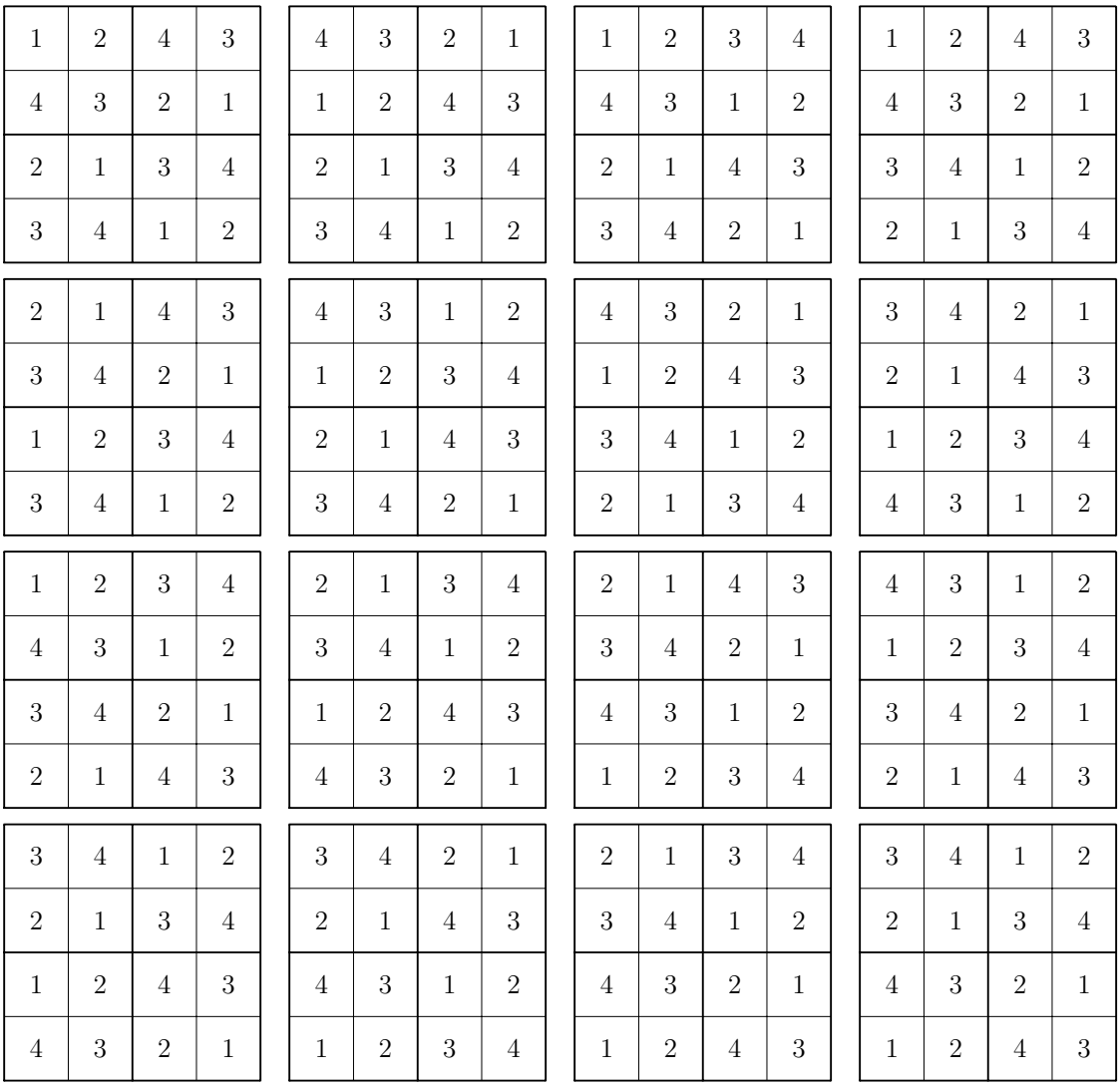


Fig. 60. 9-2 12-23 11-2 10-2 9-7 10-23 11-23 10-18 12-2 10-7 12-7 9-23 12-18 9-18 11-7 11-18

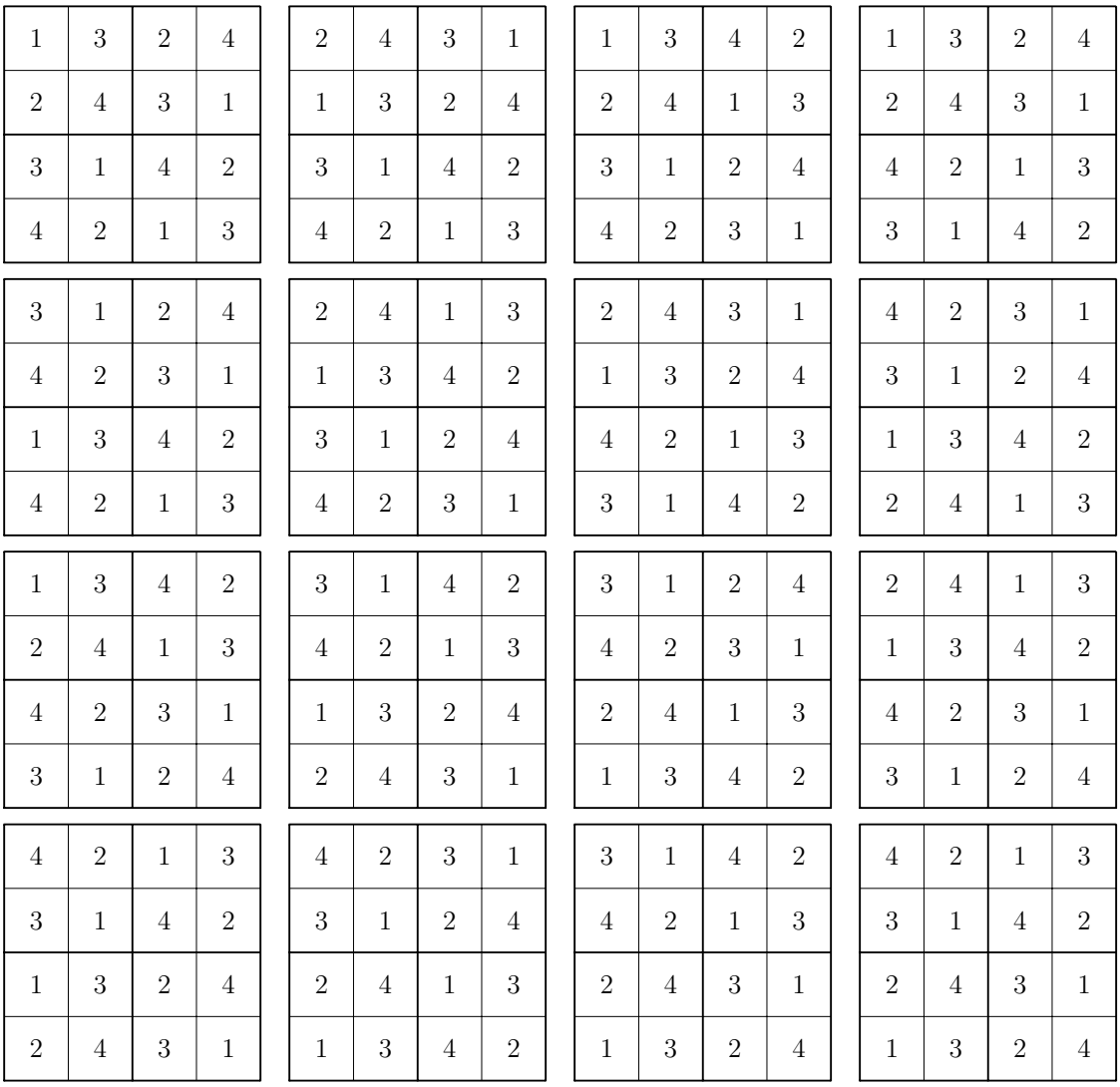


Fig. 61. 9-3 12-11 11-3 10-3 9-14 10-11 11-11 10-22 12-3 10-14 12-14 9-11 12-22 9-22 11-14 11-22

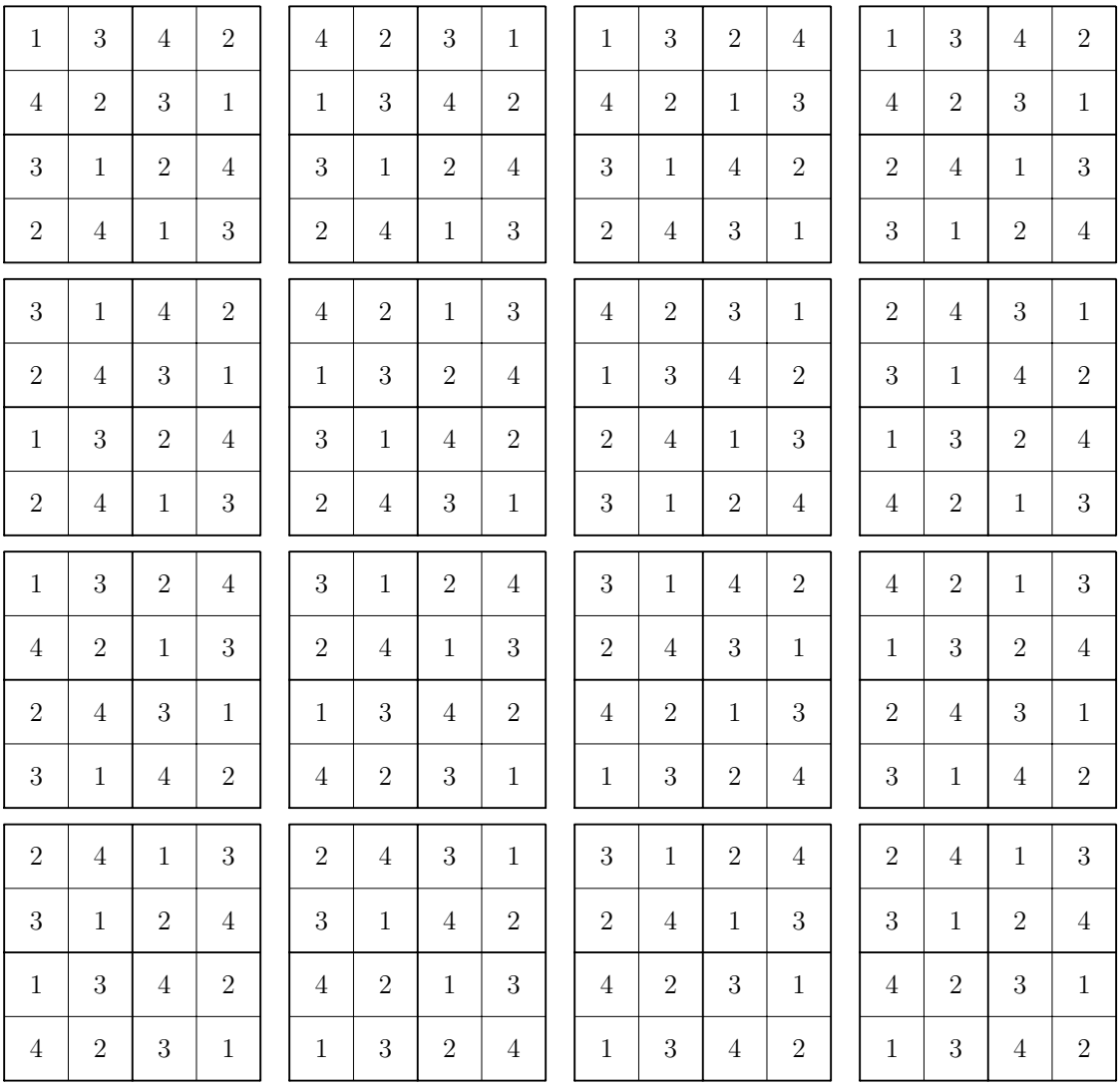


Fig. 62. 9-4 12-21 11-4 10-4 9-13 10-21 11-21 10-12 12-4 10-13 12-13 9-21 12-12 9-12 11-13 11-12

1	4	2	3
2	3	4	1
4	1	3	2
3	2	1	4

2	3	4	1
1	4	2	3
4	1	3	2
3	2	1	4

1	4	3	2
2	3	1	4
4	1	2	3
3	2	4	1

1	4	2	3
2	3	4	1
3	2	1	4
4	1	3	2

4	1	2	3
3	2	4	1
1	4	3	2
3	2	1	4

2	3	1	4
1	4	3	2
4	1	2	3
3	2	4	1

2	3	4	1
1	4	2	3
3	2	1	4
4	1	3	2

3	2	4	1
4	1	2	3
1	4	3	2
2	3	1	4

1	4	3	2
2	3	1	4
3	2	4	1
4	1	2	3

4	1	3	2
3	2	1	4
1	4	2	3
2	3	4	1

4	1	2	3
3	2	4	1
2	3	1	4
1	4	3	2

2	3	1	4
1	4	3	2
3	2	4	1
4	1	2	3

3	2	1	4
4	1	3	2
1	4	2	3
2	3	4	1

3	2	4	1
4	1	2	3
2	3	1	4
1	4	3	2

4	1	3	2
3	2	1	4
2	3	4	1
1	4	2	3

3	2	1	4
4	1	3	2
2	3	4	1
1	4	2	3

Fig. 63. 9-5 12-9 11-5 10-5 9-20 10-9 11-9 10-16 12-5 10-20 12-20 9-9 12-16 9-16 11-20 11-16

1	4	3	2
3	2	4	1
4	1	2	3
2	3	1	4

3	2	4	1
1	4	3	2
4	1	2	3
2	3	1	4

1	4	2	3
3	2	1	4
4	1	3	2
2	3	4	1

1	4	3	2
3	2	4	1
2	3	4	1
4	1	2	3

4	1	3	2
2	3	4	1
1	4	2	3
2	3	1	4

3	2	1	4
1	4	2	3
4	1	3	2
2	3	4	1

3	2	4	1
1	4	3	2
2	3	1	4
4	1	2	3

2	3	4	1
4	1	3	2
2	3	1	4
1	4	2	3

1	4	2	3
3	2	1	4
2	3	4	1
4	1	3	2

4	1	3	2
2	3	1	4
1	4	2	3
3	2	4	1

4	1	3	2
2	3	4	1
1	4	3	2
3	2	4	1

3	2	4	1
1	4	3	2
2	3	1	4
4	1	2	3

2	3	1	4
4	1	2	3
1	4	3	2
3	2	4	1

2	3	4	1
4	1	3	2
3	2	1	4
1	4	2	3

4	1	2	3
2	3	1	4
3	2	4	1
1	4	3	2

2	3	1	4
4	1	2	3
3	2	4	1
1	4	3	2

Fig. 64. 9-6 12-15 11-6 10-6 9-19 10-15 11-15 10-10 12-6 10-19 12-19 9-15 12-10 9-10 11-19 11-10

APPENDIX B

```
%Matlab code to enumerate all possible error patterns
%for the symmetric and assymetric arrays

n = 2;
num_error_patterns = 2^(n^4);

M1 = [ 1 2 3 4; 3 4 1 2; 2 1 4 3; 4 3 2 1 ];
M2 = [ 1 2 4 3; 3 4 1 2; 2 1 3 4; 4 3 2 1 ];

errors = zeros(num_error_patterns,2,2);

for i = 1:num_error_patterns

error_pattern = ones(1,n^4);
error_pattern(find(de2bi(i-1)==1)) = 0;
error_pattern = vec2mat(error_pattern,4);

S1 = solver(n,M1.*error_pattern);
S2 = solver(n,M2.*error_pattern);

size_S1 = size(S1);
size_S2 = size(S2);
```

```
23
24          if(length(size_S1) == 3)
25
26          errors(i,1,1) = 1;
27          errors(i,1,2) = size_S1(3);
28
29          end
30
31          if(length(size_S2) == 3)
32
33          errors(i,2,1) = 1;
34          errors(i,2,2) = size_S2(3);
35
36          end
37
38 end
39
40 save('error_enumeration_data','errors')
```

APPENDIX C

```
%Matlab Code to determine the distribution of path lengths
%for N randomly drawn permutations

N = 10^4;

path_length = zeros(1,N);
Sigma = cell(1,N);

for i = 1:N

        sigma = randperm(16);
        Sigma(i) = {sigma};

        index = 1;
        tau = [];
        alphabet = cell(1,16);
        alphabet(1) = {1};

        path_length(i) = 1/72*avg_path_search(sigma,tau,
            alphabet,index,0);

end
```

```
22
23 mu = sum(path_length)/length(path_length);
24
25 save('avg_path_data','Sigma','path_length','mu')
```

```
 1 function path_length = avg_path_search(sigma,tau,alphabet,
     index,path_length)
 2
 3 current_alphabet = cell2mat(alphabet(index));
 4
 5 if length(current_alphabet) == 0
 6
 7         if index == 1;
 8
 9         return
10
11         else
12
13         tau(index) = [];
14         index = index-1;
15
16         end
17 else
18
19 tau(index) = current_alphabet(1);
20 M = sudoku_matrix_generator(2,sigma,tau);
```

```
21 C = solver(2,M);
22 [N c] = entry_density(C);
23
24          if c == 1
25
26          path_length = path_length + length(tau);
27          current_alphabet(1) = [];
28          alphabet(index) = {current_alphabet};
29
30          else
31
32          current_alphabet(1) = [];
33          alphabet(index) = {current_alphabet};
34          index = index + 1;
35          alphabet(index) = {find( N(sigma(index) + [0 16 32
               48]))};
36
37          end
38 end
39
40 path_length = avg_path_search(sigma,tau,alphabet,index,
      path_length);
41
42 end
```

CONTACT INFORMATION

Name:	Andrew Young
Professional Address:	c/o Dr. Costas Georghiades Electrical and Computer Engineering Department Zachry 216B Texas A&M University College Station TX, 77840
Email Address:	ajy713@gmail.com
Education:	B.S., Electrical Engineering and Mathematics, Texas A&M University, May 2012 Summa Cum Laude Undergraduate Research Fellow